HITACHI MICROCOMPUTER SYSTEM
HMCS6800

CMOS 8-BIT SINGLE-CHIP MICROCOMPUTER

HD6305X,HD6305Y,HD63P05Y

USER'S MANUAL

HITACHI

HMCS6800

HITACHI MICROCOMPUTER SYSTEM

CMOS 8-BIT SINGLE-CHIP MICROCOMPUTER

HD6305X,HD6305Y,HD63P05Y

USER'S MANUAL

HITACHI

68-1-141

MAR. 1986 68-1-141

Circuit examples and characteristics examples described in this manual are used to explain typical application examples of the Hitachi Semiconductors.

Pay attention to the following points in using this manual.

1. The contents of this manual may be changed without prior notice.
2. A part or all of this manual may not be reproduced or republished in any form without prior permission of the company.
3. The company is not responsible for any damage that may be caused by an accident on the users' side.
4. This manual does not provide any guarantee of implementation of industrial ownership on other right or approval of implementation right.

CONTENTS

1. OVERVIEW 1

1.1 Features 1
1.2 Block Diagram 5

2. INTERNAL HARDWARE AND OPERATIONS 13

2.1 Memory 13
2.2 Registers 15
2.3 Timer 17
2.4 Serial Communication Interface (SCI) 20
2.5 Reset 25
2.6 Internal Oscillator Options 25
2.7 Interrupts 27
2.8 Input/Output Ports 30
2.9 Low Power Dissipation Mode 32
2.10 Bit Manipulation 37
2.11 Addressing Modes 38
2.12 Instruction Set 43
2.13 Operation at Each Instruction Cycle 50

3. INSTRUCTION SYSTEM 54

3.1 Symbols and Abbreviations 54
3.2 Executable Instruction 56

4. APPLICATIONS AND PRECAUTIONS 121

4.1 Memory Space Expansion of HD6305X1/Y1, HD6305X2/Y2, HD63P05Y1 121
4.2 Watch-Dog Timer 122
4.3 Auto Reset Circuit 123
4.4 Manual Reset Circuit 125
4.5 A/D Converter Circuit (1) ... High Speed 126
4.6 A/D Converter Circuit (2) ... Low Speed 128
4.7 Precaution;- Board Design of Oscillation Circuit 129
4.8 Precaution; - Program of Write Only Register 130
4.9 Precaution; - Sending/Receiving Program of Serial Data 130
4.10 Precaution; - WAIT/STOP Instructions Program 131
4.11 Precaution; - To use the EPROM ON-PACKAGE 8-bit Single-chip Microcomputer 131

5. PIN ARRANGEMENT AND DIMENSIONAL OUTLINE 133

6. ELECTRICAL CHARACTERISTICS 136

6.1 Electrical Characteristics of HD6305X0, HD6305Y0, HD63P05Y0... 136

6.2 Electrical Characteristics of HD6305X1/X2, HD6305Y1/Y2, HD63P05Y1 141

7. ROM CODE ORDER METHOD 148

<APPENDIX>

I. DESIGN PROCEDURE AND SUPPORT TOOL 149

II. ORDERING SPECIFICATIONS..... 151

1. OVERVIEW

1.1 Features

The HD6305X0, HD6305X1, HD6305X2 MCU are CMOS 8-bit single chip Microcomputers containing CPU, Clock Oscillator, ROM (HD6305X2 does not contain Internal ROM), RAM, I/O port, Timer and Serial Communication Interface (SCI) on a single chip. The HD6305X1 and HD6305X2 MCU configurations are equivalent to the HD6305X0 MCU configuration except the number of I/O terminals.

The HD6305X0 and HD6305X1 MCU contain 4096 bytes of Internal ROM. The HD6305X1 MCU expands its memory up to 12k-bytes externally. The HD6305X2 MCU, without Internal ROM, expands its memory space up to 16k bytes.

Features of these Microcomputers are as follows.

- Upward Compatible with the HD6805 Family Instructions
- 8-bit Architecture
- 4096-Bytes of Internal ROM (HD6305X0)
- 4096-Bytes of Internal ROM and 12k-Bytes of External Memory (HD6305X1)
- 16K-Bytes of External Memory (HD6305X2)
- 128-Bytes of RAM
- I/O Terminal × 32, Input Only Terminal × 7, Output Only Terminal × 16 (HD6305X0)

  I/O Terminal × 24, and Input Only Terminal × 7 (HD6305X1, HD6305X2)
- 2 Internal Timers

  An 8-Bit Timer with 7-Bit Prescaler × 1

  A 15-Bit Timer (Multiplexed with SCI Clock Driver) × 1
- Internal Serial Communication Interface
- Interrupts; External Interrupt × 2, Timer Interrupt × 2, SCI Interrupt × 1, Soft Interrupt × 1
- Low Power Dissipation Modes

  Wait Mode; Continues clock oscillation, stops CPU, permits Timer/Serial/Interrupt to function

  Stop Mode; Stops clock, holds RAM data I/O Status, and Register contents

  Standby Mode; Stops clock, holds RAM data, resets internal options.

- Minimum Instruction Cycle Time
  HD6305X0/X1/X2 .... 1 µs (f= 1MHz)
  HD63A05X0/X1/X2 ... 0.67 µs (f= 1.5MHz)
  HD63B05X0/X1/X2 ... 0.5 µs (f= 2MHz)
- Wide Operation Range
  V_{CC} = 3 ∿ 6V (f=0.1 ∿ 0.5MHz)
  HD6305X0/X1/X2 f=0.1 ∿ 1MHz (V_{CC}=5V±10%)
  HD63A05X0/X1/X2 f=0.1 ∿ 1.5MHz (V_{CC}=5V±10%)
  HD63B05X0/X1/X2 f=0.1 ∿ 2MHz (V_{CC}=5V±10%)
- Sufficient Supporting System by Evaluation Kit
  3 Additional Instructions ... DAA, WAIT, STOP

The HD6305Y0, HD6305Y1, HD6305Y2 MCU are CMOS 8-bit single chip Microcomputers containing CPU, Clock Oscillator, ROM (HD6305Y2 does not contain Internal ROM), RAM, I/O port, Timer, and Serial Communication Interface (SCI) on a single chip. The HD6305Y1 and HD6305Y2 MCU configurations are equivalent to the HD6305Y0 MCU configuration except the number of I/O terminals.

The HD6305Y0 and HD6305Y1 MCU contain 7872 bytes of Internal ROM. The HD6305Y1 MCU expands its memory space up to 8k-bytes externally. The HD6305Y2 MCU, without Internal ROM, expands its memory space up to 16k bytes externally.

Features of these Microcomputers are as follows.

- Upward Compatible with the HD6805 Family Instructions
- 8-bit Architecture
- 7872-Bytes of Internal ROM (HD6305Y0)
- 7872-Bytes of Internal ROM and 8k-Bytes of External Memory (HD6305Y1)
  16k-Bytes of External Memory (HD6305Y2)
- 256-Bytes of RAM
- I/O Terminal × 32, Input Only Terminal × 7, Output Only Terminal × 16 (HD6305Y0)
  I/O Terminal × 24, and Input Only Terminal × 7 (HD6305Y1, HD6305Y2)
- 2 Internal Timers
  An 8-Bits Timer with 7-Bits Prescaler × 1
  A 15-Bits Timer (Multiplexed with SCI Clock Driver) × 1
- Internal Serial Communication Interface

- Interrupts; External Interrupt × 2, Timer Interrupt × 2,
  SCI Interrupt × 1, Soft Interrupt × 1.
- Low Power Dissipation Modes
  Wait Mode; Continues clock oscillation, stops CPU, permits Timer/Serial/Interrupt to function.
  Stop Mode; Stops clock, holds RAM data, I/O status, and Register contents.
  Standby Mode; Stops clock, holds RAM data, resets internal options.
- Minimum Instruction Cycle Time
  HD6305Y0/Y1/Y2 .... 1 µs (f= 1MHz)
  HD63A05Y0/Y1/Y2 ... 0.67 µs (f= 1.5MHz)
  HD63B05Y0/Y1/Y2 ... 0.5 µs (f= 2MHz)
- Wide Operation Range
  V_{CC} = 3 ∿ 6V (f=0.1 ∿ 0.5MHz)
  HD6305Y0/Y1/Y2 f=0.1 ∿ 1MHz (V_{CC}=5V±10%)
  HD63A05Y0/Y1/Y2 f=0.1 ∿ 1.5MHz (V_{CC}=5V±10%)
  HD63B05Y0/Y1/Y2 f=0.1 ∿ 2MHz (V_{CC}=5V±10%)
- Sufficient Supporting System by Evaluation Kit
  3 Additional Instructions ... DAA, WAIT, STOP

The HD6305 family instruction sets, which are completely compatible, are available for system expansion.

Table 1-1 and 1-2 summarizes the features of each LSI.

Table 1-1 Features of HD6305X, HD6305Y

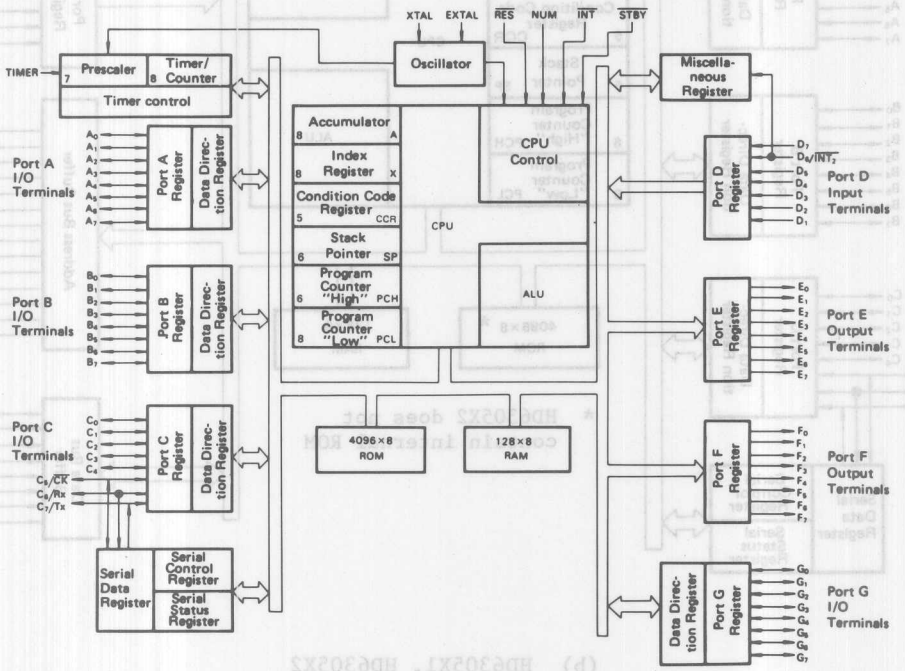
Type No.		HD6305X0 HD63A05X0 HD63B05X0	HD6305X1 HD63A05X1 HD63B05X1	HD6305X2 HD63A05X2 HD63B05X2	HD6305Y0 HD63A05Y0 HD63B05Y0	HD6305Y1 HD63A05Y1 HD63B05Y1	HD6305Y2 HD63A05Y2 HD63B05Y2
Package		DP-64S/FP-64	DP-64S/FP-64	DP-64S/FP-64	DP-64S/FP-64	DP-64S/FP-64	DP-64S/FP-64
Internal Memory	ROM(k bytes)	4	4	–	7.9	7.9	–
	RAM (bytes)	128	128	128	256	256	256
External Memory	(Expanded) (K bytes)	–	12	16	–	8	16
I/O Port	Input Output	55 / 32	31 / 24	31 / 24	55 / 32	31 / 24	31 / 24
	Input	7	7	7	7	7	7
	Output	16	–	–	16	–	–
Interrupt	Nestings	12	12	12	12	12	12
	External Interrupt	2	2	2	2	2	2
	Soft Interrupt	1	1	1	1	1	1
	Timer Interrupt	2	2	2	2	2	2
	Serial Interrupt	1	1	1	1	1	1
Timer	8 bits (with 7 bits prescaler)	1	1	1	1	1	1
	15 bits	1	1	1	1	1	1
Oscillator	Crystal	Possible	Possible	Possible	Possible	Possible	Possible
	Ceramic Oscillator	Possible	Possible	Possible	Possible	Possible	Possible
EPROM on-package type		HD63P05Y0 HD63PA05Y0 HD63PB05Y0	HD63P05Y1 HD63PA05Y1 HD63PB05Y1	—	HD63P05Y0 HD63PA05Y0 HD63PB05Y0	HD63P05Y1 HD63PA05Y1 HD63PB05Y1	—

Table 1-2 Features of EPROM On-Package Type Family

Type No.		HD63P05Y0	HD63PA05Y0	HD63PB05Y0	HD63P05Y1	HD63PA05Y1	HD63PB05Y1
Package		DC-64SP	DC-64SP	DC-64SP	DC-64SP	DC-64SP	DC-64SP
Equivalent Device		HD6305X0 HD6305Y0	HD63A05X0 HD63A05Y0	HD63B05X0 HD63B05Y0	HD6305X1 HD6305Y1	HD63A05X1 HD63A05Y1	HD63B05X1 HD63B05Y1
EPROM	4 k bytes	HN482732A-30	HN482732A-30	HN482732A-25	HN482732A-30	HN482732A-30	HN482732A-25
	8 k bytes	HN482764-3 HN27C64-30	HN482764-3 HN27C64-30	HN482764 HN27C64-25	HN482764-3 HN27C64-30	HN482764-3 HN27C64-30	HN482764 HN27C64-25

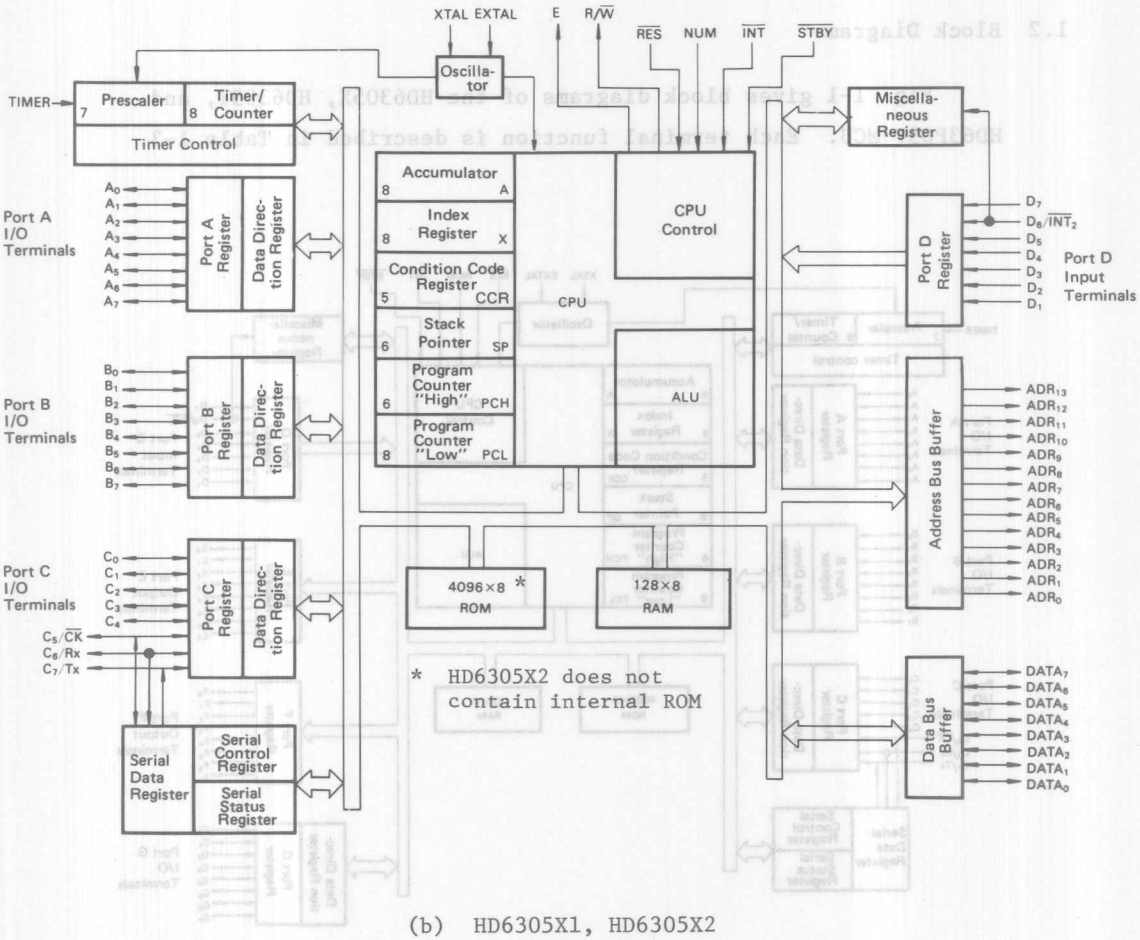
1.2 Block Diagram

Fig. 1-1 gives block diagrams of the HD6305X, HD6305Y, and HD63P05Y MCU. Each terminal function is described in Table 1-3.



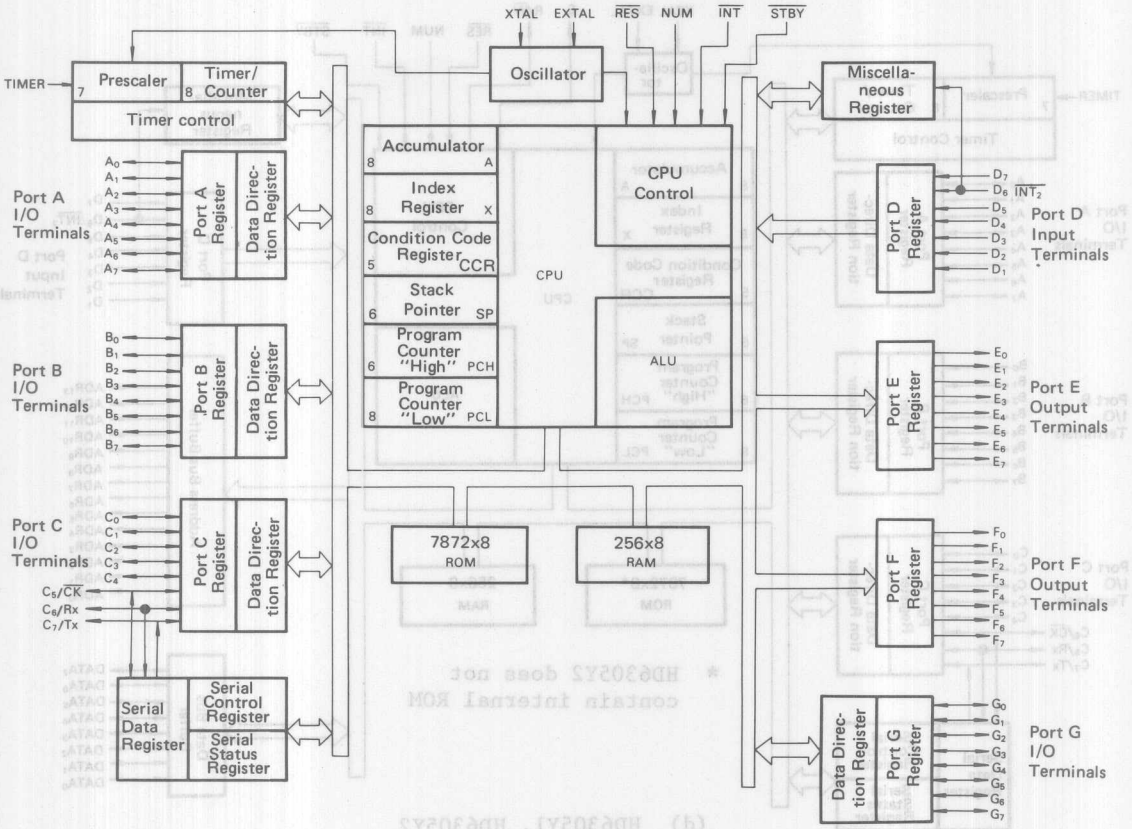
(a) HD6305X0

Fig. 1-1 Block Diagram (to be continued)



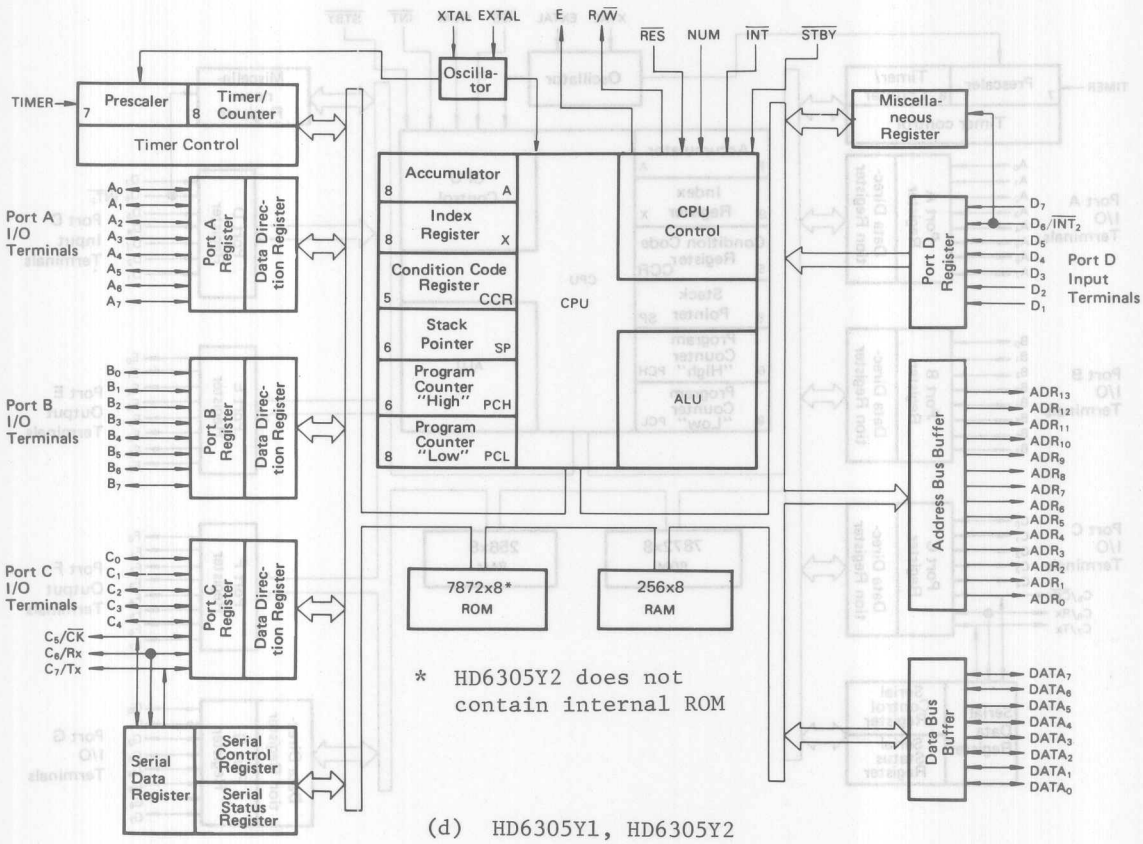
(b) HD6305X1, HD6305X2

Fig. 1-1 Block Diagram (to be continued)



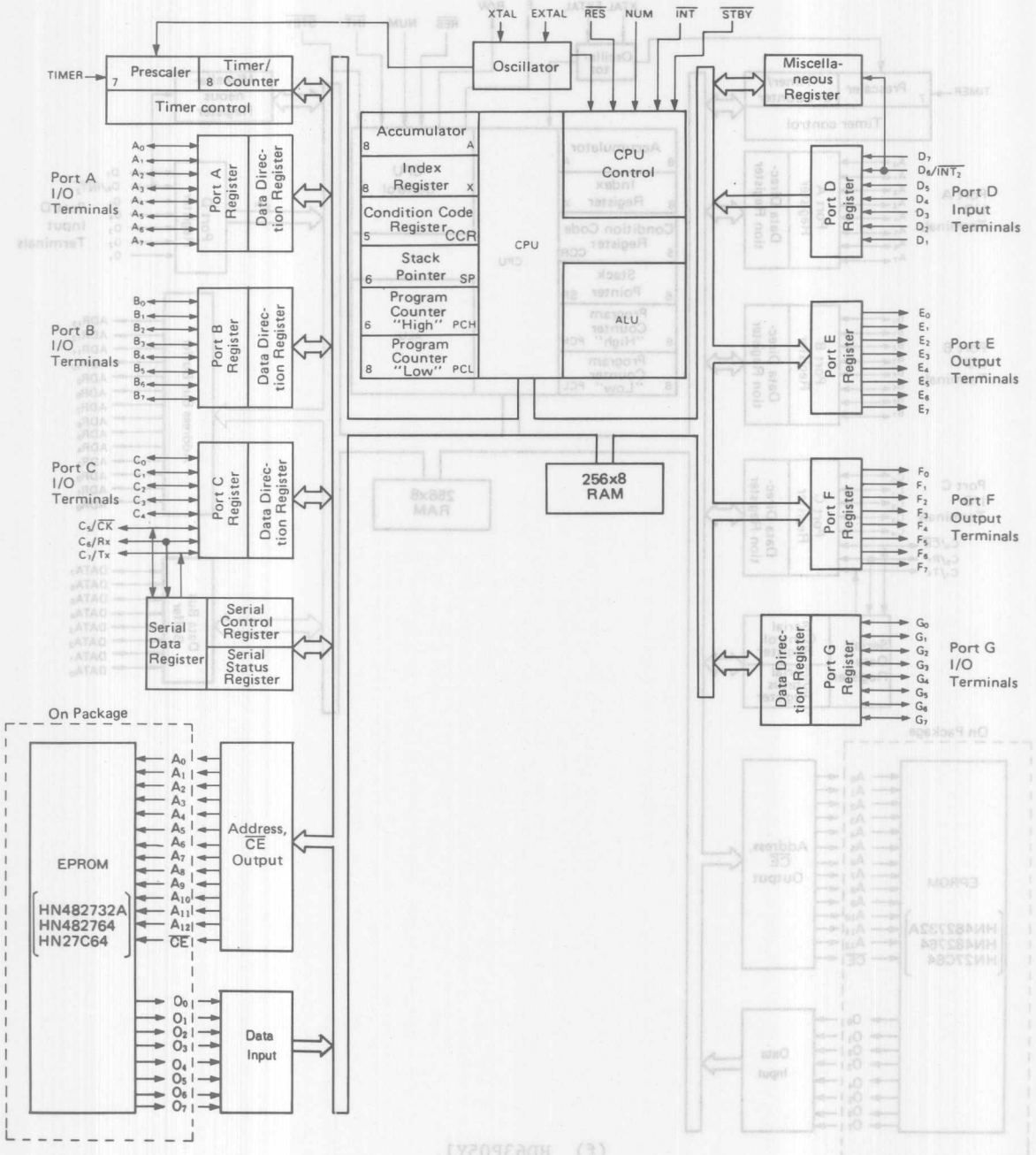
(c) HD6305YO

Fig. 1-1 Block Diagram(to be continued)



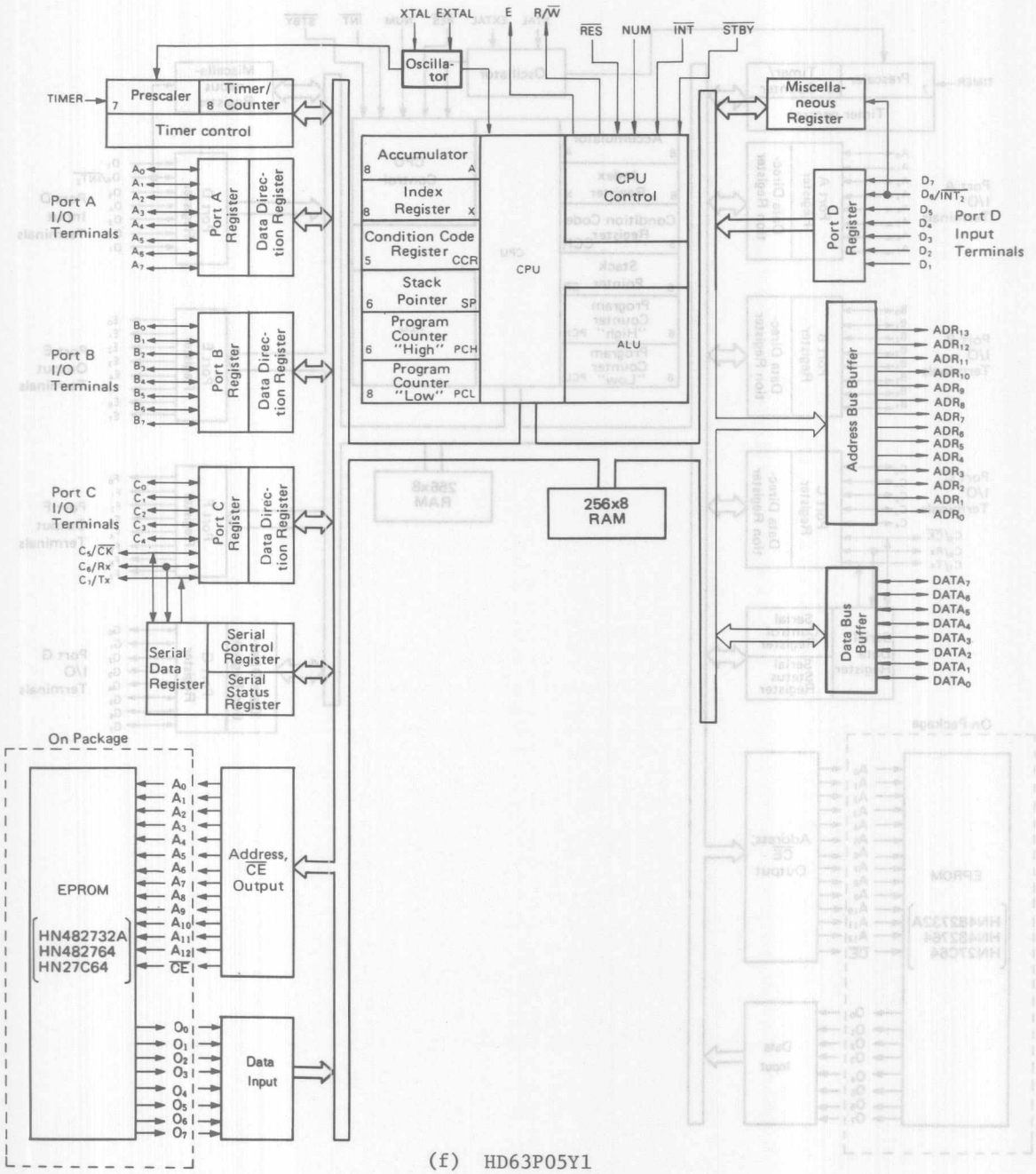
(d) HD6305Y1, HD6305Y2

Fig. 1-1 Block Diagram (to be continued)



(e) HD63P05Y0

Fig. 1-1 Block Diagram (to be continued)



(f) HD63P05Y1

Fig. 1-1 Block Diagram

Table 1-3 Terminal Functions

Terminals	Functions	Corresponding Devices
V_{CC}, Vss	Voltage is applied to the MCU through these terminals. V_{CC} provides 5.0V±10% power supply, while Vss is grounded.	
$\overline{INT}$, $\overline{INT_2}$	These terminals function as external interrupt request inputs to the MCU. Refer to "2.7 INTERRUPT" for details. The $\overline{INT_2}$ is multiplexed with the D_6.	
XTAL, EXTAL	These terminals are inputs to the internal clock circuit. Connect a crystal oscillator (AT cut, 2.0~8.0 MHz) or ceramic filter to these terminals. Refer to "2.6 INTERNAL OSCILLATOR" for further informations.	HD6305X0/Y0 HD6305X1/Y1 HD6305X2/Y2 HD63P05Y0/Y1
TIMER	This external input terminal controls internal timer circuit. Refer to "2.3 TIMER" for details.	
$\overline{RES}$	This terminal resets the MCU. Refer to "2.5 RESET" for details.	
NUM	This terminal is not for user application. Connect the HD6305X1, HD6305Y1 and HD63P05Y1 MCU to V_{CC} through resistance 10kΩ. Ground HD6305X0, HD6305X2, HD6305Y0, HD6305Y2 and HD63P05Y0 MCU to Vss.	
Enable(E)	This terminal transmits E clock. This output is of single phase and TTL compatible, and has a clock frequency which is a quarter of crystal oscillation frequency or external clock frequency. This terminal drives one TTL load and 90pF capacitance.	HD6305X1/Y1 HD6305X2/Y2 HD63P05Y1

(to be continued)

Terminals	Functions	Corresponding Devices
Port A I/O terminals (A_0~A_7) Port B I/O terminals (B_0~B_7) Port C I/O terminals (C_0~C_7) Port G I/O terminals (G_0~G_7)	Each terminal can be individually programmed as an input or as an output by programming the Data Direction Register. Refer to "2.8 I/O Port" for details.	HD6305X0/Y0 HD6305X1/Y1 HD6305X2/Y2 HD63P05Y0/Y1 (HD6305X1/Y1, HD6305X2/Y2, HD63P05Y1 don't contain G Port I/O terminal)
Port D Input terminals (D_1~D_7)	These 7 terminals are TTL/CMOS compatible Input Only terminals. D_6 of Port D is multiplexed with $\overline{INT_2}$. When D_6 functions as a port, set $\overline{INT_2}$ interrupt mask bit of Miscellaneous Register to "1" to mask $\overline{INT_2}$ interrupt.	HD6305X0/Y0 HD6305X1/Y1 HD6305X2/Y2 HD63P05Y0/Y1
Port E Output terminals (E_0~E_7) Port F Output terminals (F_0~F_7)	These 16 terminals are TTL/CMOS compatible Output only terminals.	HD6305X0/Y0 HD63P05Y0
Data bus ($DATA_0$~$DATA_7$)	These 8 I/O terminals are for Data Bus. They drive one TTL load and 90pF capacitance.	HD6305X1/Y1 HD6305X2/Y2 HD63P05Y1
Address bus (ADR_0~ADR_{13})	These 14 output only terminals are for Address Bus. They drive one TTL load and 90pF capacitance.	
$\overline{STBY}$	This terminal permits the MCU to enter into the Standby Mode. When $\overline{STBY}$ goes into "Low" level, the oscillation stops and the MCU enters into the Reset Mode. Refer to "Standby Mode" in "2.9 LOW POWER CONSUMPTION MODE" for details.	HD6305X0/Y0 HD6305X1/Y1 HD6305X2/Y2 HD63P05Y0/Y1
$\overline{CK}$ (C_5)	This terminal transmits or receives SCI clock for SCI operation. Refer to "2.4 SERIAL COMMUNICATION INTERFACE" for details.	
Rx (C_6)	This terminal receives serial data. Refer to "2.4 SERIAL COMMUNICATION INTERFACE" for details.	
Tx (C_7)	This terminal transmits serial data. Refer to "2.4 SERIAL COMMUNICATION INTERFACE" for details.	

2. INTERNAL HARDWARE AND OPERATIONS

2.1 Memory

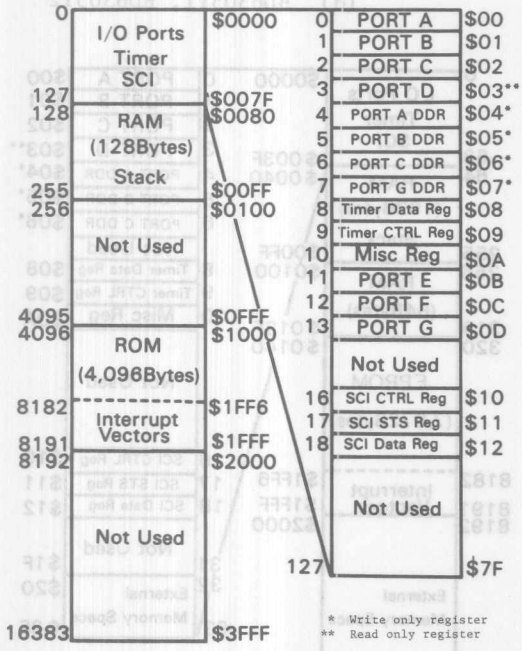
Fig. 2-1 shows the memory map for the HD6305X, HD6305Y, HD63P05Y MCU.

When an interrupt occurs, the CPU register contents are pushed onto the stack in the order as Fig. 2-2 shows, because stack pointer is decreased during pushes.

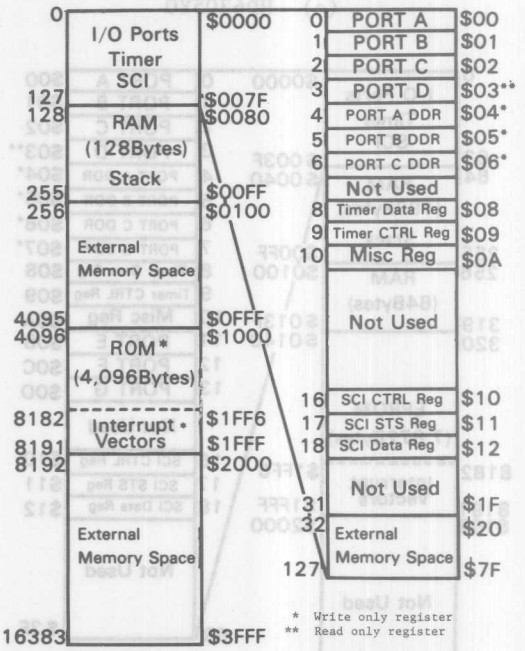
The lower byte (PCL) of the Program Counter, the higher byte (PCH) of the Program Counter, Index Register (IX), Accumulator (A) and Condition Code Register are stacked in this order.

In a subroutine call, only the Program Counter (PCL, PCH) is pushed onto the stack.

Note that the Stack Pointer specifies the next stack area to store data and decreased after stacking 2-byte data. Meanwhile, the Stack Pointer is increased before pulling 1-byte from stack.



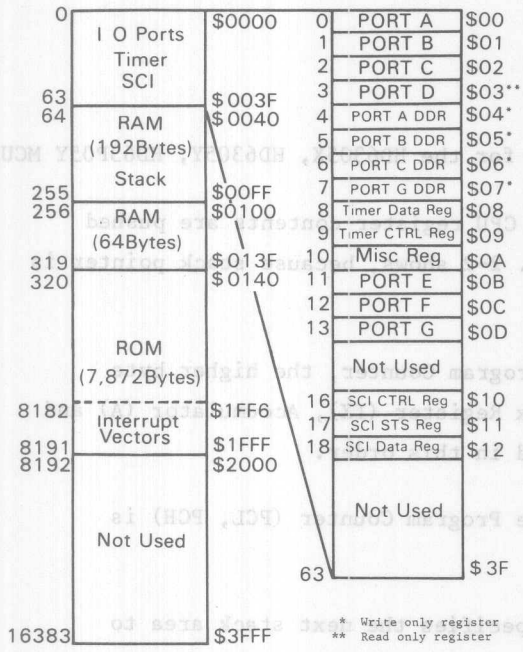
(a) HD6305X0



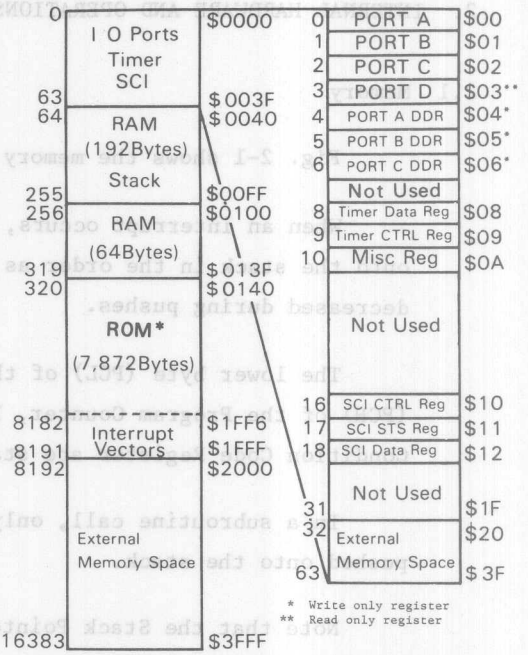
* ROM area ($1000 ∿ $1FFF) in the HD6305X2 changes into External Memory Space.

(b) HD6305X1, HD6305X2

Fig. 2-1 Memory Map

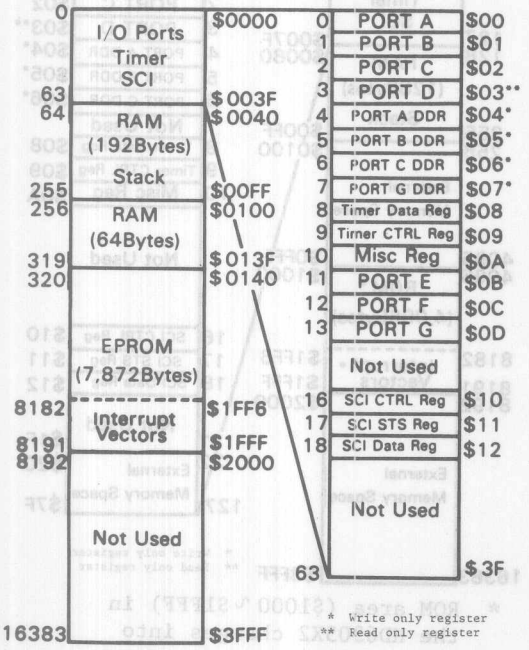


(c) HD6305Y0

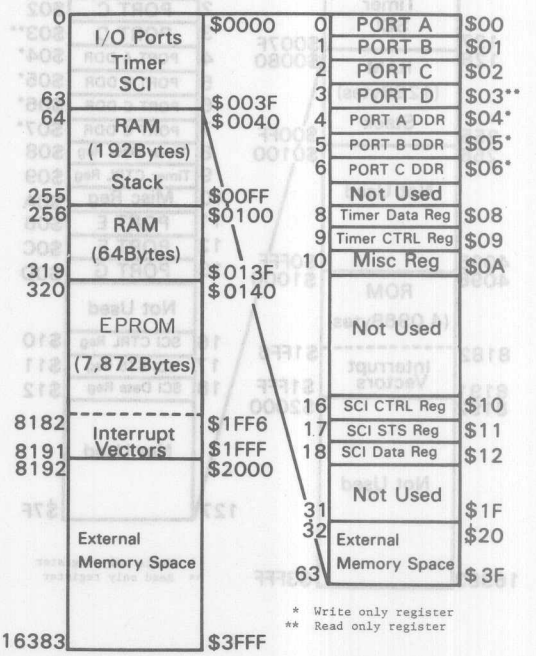


* ROM area ($0140 ∿ $1FFF) in the HD6305Y2 changes into External Memory Space.

(d) HD6305Y1, HD6305Y2



(e) HD63P05Y0



(f) HD63P05Y1

Fig. 2-1 Memory Map

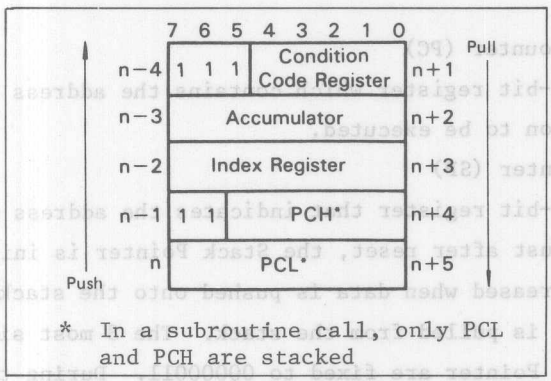


Fig. 2-2 Sequence of Interrupt Stacking

2.2 Registers

CPU has 5 registers available to programmers. Fig. 2-3 shows these.

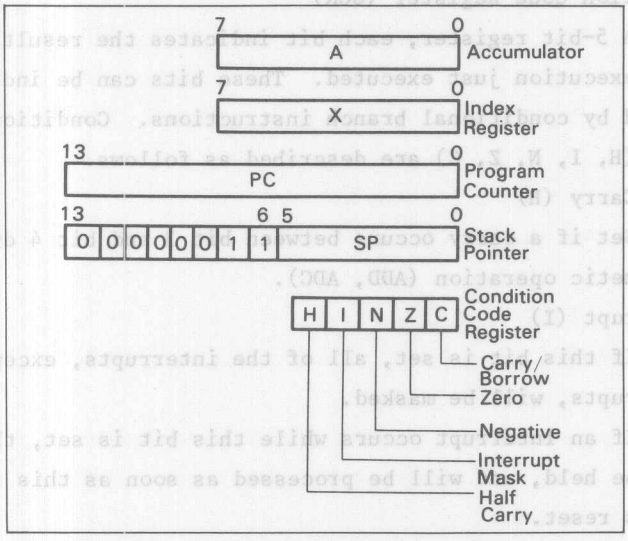


Fig. 2-3 Programming Model

(1) Accumulator (A)

An 8-bit general purpose register to hold operands, and results of arithmetic operations and data processings.

(2) Index Register (X)

An 8-bit register for the Indexed Addressing Mode. An 8-bit address contained in the Index Register will create an effective address if an offset value is added to it.

When executing a read/modify/write instruction, the Index Register is available for holding operands or the result of operation instead of the Accumulator.

The Index Register is also available for storing data temporarily.

(3) Program Counter (PC)

A 14-bit register which contains the address of the next instruction to be executed.

(4) Stack Pointer (SP)

A 14-bit register that indicates the address of the next stack space. Just after reset, the Stack Pointer is initialized to $00FF. It is decreased when data is pushed onto the stack, and is increased when data is pulled from the stack. The 8 most significant bits of the Stack Pointer are fixed to 00000011. During the MCU is reset or executing Reset Stack Pointer (RSP) instruction, the Stack Pointer is set to $00FF. Since subroutines and interrupts are designed to use memory space up to address $00C1 for stacking, programmers can use up to 31 levels for subroutine, and up to 12 levels for interrupts.

(5) Condition Code Register (CCR)

A 5-bit register, each bit indicates the result of the instruction execution just executed. These bits can be individually tested by conditional branch instructions. Condition Code Register bits (H, I, N, Z, C) are described as follows.

Half Carry (H)

Set if a carry occurs between bit 3 and bit 4 during an arithmetic operation (ADD, ADC).

Interrupt (I)

If this bit is set, all of the interrupts, except software interrupts, will be masked.

If an interrupt occurs while this bit is set, the interrupt will be held, and will be processed as soon as this interrupt mask bit is reset.

After executing an instruction following to the CLI instruction, the CPU will enter into the interrupt service routine.

Negative (N)

Set if the result of the arithmetic operation, logical operation or data processing is negative (bit 7 is set to "1"); otherwise reset.

Zero (Z)

Set if the result of the arithmetic operation, logical operation, or data processing is "0".

Carry/Borrow (C)

Set if a carry or borrow occurs in the last arithmetic operation. This bit is affected by Bit Test and Branch Instruction, Shift Instruction, and Rotate Instruction.

2.3 Timer

Fig. 2-4 shows the MCU Timer Block Diagram.

8-bit Timer Data Register (TDR) is loaded by program control. When it receives the clock input, it starts counting down.

When TDR counts down to "0", the timer interrupt request bit (TCR7) in the Timer Control Register (TCR) is set to "1".

In response to the interrupt request, the CPU saves its status into the stack and fetches timer interrupt routine address from addresses $1FF8 and $1FF9 and execute the interrupt routine. The timer interrupt can be masked by setting the timer interrupt mask bit (bit 6) in the timer control register. The mask bit (I) in the condition code register can also mask the timer interrupt.

The source clock to the timer can be either an external signal from the timer input terminal or the internal E signal (the oscillator clock divided by 4). If the E signal is used as the source, the clock input can be gated by the input to the timer input terminal.

Once the timer count has reached 0, it starts counting down with "$FF". The count can be monitored whenever desired by reading the timer data register. This permits the program to know the length of time having passed after the occurrence of a timer interrupt, without disturbing the contents of the counter.

When the MCU is reset, the prescaler and counter are both initialized to logic "1". The timer interrupt request bit (bit 7) is cleared and the timer interrupt mask bit (bit 6) is set.

Write "0" in that bit to clear the timer interrupt request bit (bit 7). (Refer to Table 2-1)

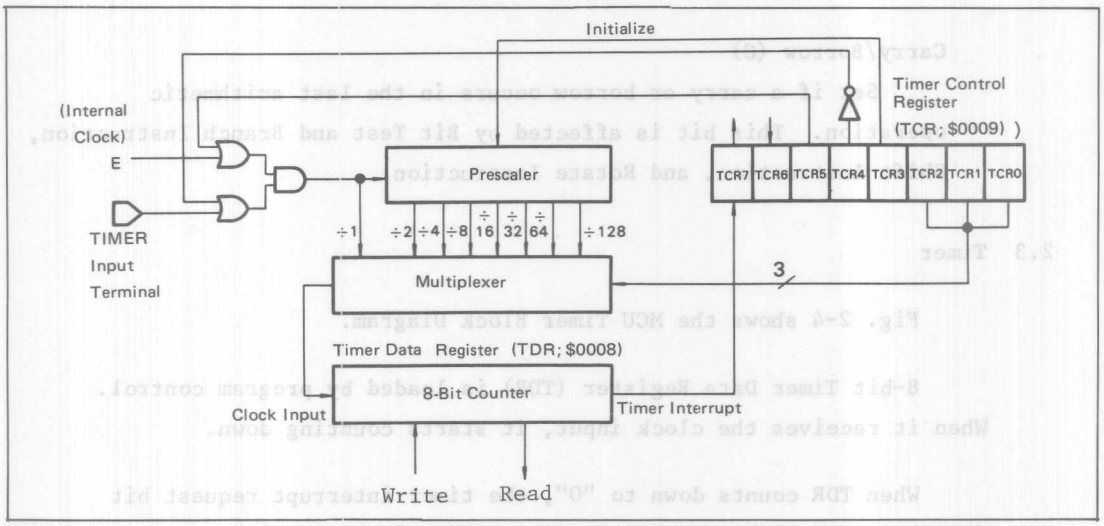


Fig. 2-4 Timer Block Diagram

Table 2-1

TCR7	Timer interrupt request
0	Absent
1	Present

TCR6	Timer interrupt mask
0	Enabled
1	Disabled

(1) Timer Control Register (TCR; $0009)

Selection of a clock source, selection of a prescaler frequency division ratio, and a timer interrupt can be controlled by the timer control register (TCR; $0009).

For the selection of a clock source, any one of the four modes (Refer to Table 2-2) can be selected by bits 5 and 4 of the timer control register (TCR).

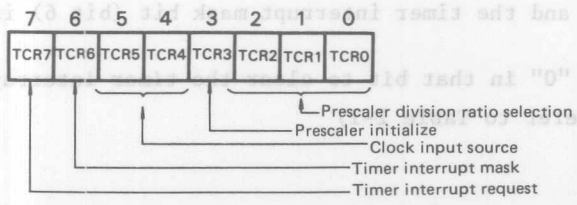


Fig. 2-5 Timer Control Register (TCR; $0009)

After reset, the TCR is initialized to "E under timer terminal control" (bit 5 = 0, bit 4 = 1). If the TIMER is set to "1", the counter starts counting down with "$FF" immediately after reset.

When "1" is written in bit 3, the prescaler is initialized. This bit is always initialized to "0" when read.

A prescaler division ratio is selected by the combination of three bits (bits 0, 1 and 2) of the timer control register (Refer to Table 2-3). There are eight different division ratios: ÷1, ÷2, ÷4, ÷8, ÷16, ÷32, ÷64 and ÷128. After reset, the TCR is set to the ÷1 mode.

A timer interrupt is enabled when the timer interrupt mask bit is "0", and disabled when the bit is "1". When a timer interrupt occurs, "1" is set in the timer interrupt request bit. This bit can be cleared by writing "0" in that bit.

Table 2-2 Clock Source Selection

TCR		Clock input source
Bit 5	Bit 4	
0	0	Internal clock E
0	1	E under TIMER terminal control
1	0	No clock input (counting stopped)
1	1	Event input from TIMER terminal

Table 2-3 Prescaler Division Ratio Selection

TCR			Prescaler division ratio
Bit 2	Bit 1	Bit 0	
0	0	0	÷1
0	0	1	÷2
0	1	0	÷4
0	1	1	÷8
1	0	0	÷16
1	0	1	÷32
1	1	0	÷64
1	1	1	÷128

2.4 Serial Communication Interface (SCI)

The SCI is used to transmit or receive 8-bit data in serial 16 types of transfer rates, in the range from 1 µs to approx. 32 ms in 4MHz operation, are available.

The SCI is consisted of 3 registers, 1 eighth counter, and 1 prescaler. (Refer to Fig. 2-6.)

The SCI communicates with the CPU through data bus, and external I/O devices through bit 3, 4 and 5 of Port C.

The following explains each register function and the SCI operation.

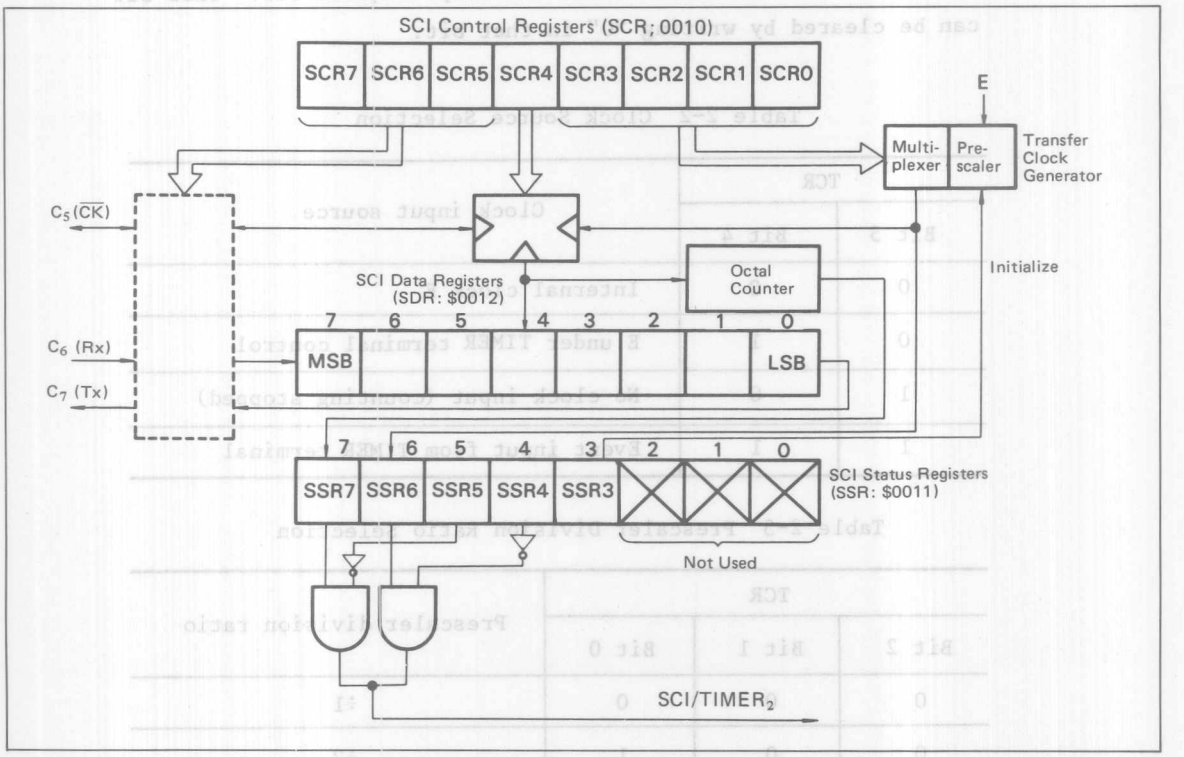


Fig. 2-6 SCI Block Diagram

(1) SCI Control Register (SCR: $0010)

Fig. 2-7 shows SCI Control Register configuration.

Bit 7 (SCR7)

When this bit is set to "1", the DDR corresponding to C_7 is set to "1" and C_7 transmits SCI data. After reset, the bit is initialized to "0".

Bit 6 (SCR6)

When this bit is set to "1", the DDR corresponding to C_6 is set to "0" and C_6 transmits SCI data. After reset, the bit is initialized to "0".

Bits 5 and 4 (SCR5, SCR4)

These bits select a clock source. After reset, the bits are initialized to "0".

Bits 3 ∿ 0 (SCR3 ∿ SCR0)

These bits select a transfer clock rate. After reset, the bits are initialized to "0".

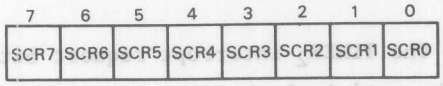


Fig. 2-7 SCI Control Register

SCR3	SCR2	SCR1	SCR0	Transfer clock rate	
				4.00 MHz	4.194 MHz
0	0	0	0	1 µs	0.95 µs
0	0	0	1	2 µs	1.91 µs
0	0	1	0	4 µs	3.82 µs
0	0	1	1	8 µs	7.64 µs
∿	∿	∿	∿	∿	∿
1	1	1	1	32768 µs	1/32s

SCR7	C_7 terminal
0	Used as I/O terminal (by DDR).
1	Serial data output (DDR output)

SCR6	C_6 terminal
0	Used as I/O terminal (by DDR).
1	Serial data input (DDR input)

SCR5	SCR4	Clock source	C_5 terminal
0	0	-	Used as I/O terminal (by DDR).
0	1	-	
1	0	Internal	Clock output (DDR output)
1	1	External	Clock input (DDR input)

(2) SCI Data Register (SDR; $0012)

A serial-parallel conversion register that is used for transferring data.

(3) SCI Status Register (SSR; $0011)

Bit 7 (SSR7)

Bit 7 is the SCI interrupt request bit which is set upon completion of transmitting or receiving 8-bit data. It is cleared when reset or data is written to or read from the SCI data register with the SCR5="1". The bit can also be cleared by writing "0" into it.

Bit 6 (SSR6)

Bit 6 is the $TIMER_2$ interrupt request bit. $TIMER_2$ is multiplexed with the serial clock generator, and SSR6 is set each time the internal transfer clock falls. When reset, the bit is cleared. It also be cleared by writing "0" in it. (For details, see $TIMER_2$.)

Bit 5 (SSR5)

Bit 5 is the SCI interrupt mask bit which can be set or cleared by software. When this bit is set to "1", the SCI interrupt (SSR7) is masked. When reset, it is set to "1".

Bit 4 (SSR4)

Bit 4 is the $TIMER_2$ interrupt mask bit which can be set or cleared by software. When the bit is set to "1", the $TIMER_2$ interrupt (SSR6) is masked. When reset, it is set to "1".

Bit 3 (SSR3)

When "1" is written into this bit, the prescaler of the transfer clock generator is initialized. When read, the bit is always "0".

Bits 2 ∿ 0

Not used.

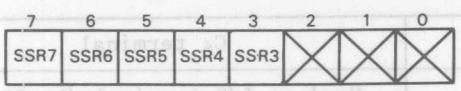


Fig. 2-8 SCI Status Register (SSR; $0011)

SSR7	SCI interrupt request
0	Absent
1	Present

SSR6	$TIMER_2$ interrupt request
0	Absent
1	Present

SSR5	SCI interrupt mask
0	Enabled
1	Disabled

SSR4	$TIMER_2$ interrupt mask
0	Enabled
1	Disabled

(4) Data Transmission

By writing the desired control bits into the SCI control registers, a transfer rate and a source of transfer clock are determined and bits 7 and 5 of port C are set at the serial data output terminal and the serial clock terminal, respectively. The transmit data should be stored from the accumulator or index register into the SCI data register. The data written in the SCI data register is transmitted from the C_7/Tx terminal, starting with the LSB, synchronously with the falling edge of the serial clock. (Refer to Fig. 2-9.) When 8 bits of data have been transmitted, the interrupt request bit is set in bit 7 of the SCI status register with the rising edge of the last serial clock. This request can be masked by setting bit 5 of the SCI status register. Once the data has been transmitted, the 8th bit data (MSB) stays at the C_7/Tx terminal. If an external clock source has been selected, the transfer rate determined by bits 0 ∿ 3 of the SCI control register is ignored, and the $C_5/\overline{CK}$ terminal is set as input. If the internal clock has been selected, the $C_5/\overline{CK}$ terminal is set as output and clocks are transmitted at the transfer rate selected by bits 0 ∿ 3 of the SCI control register.

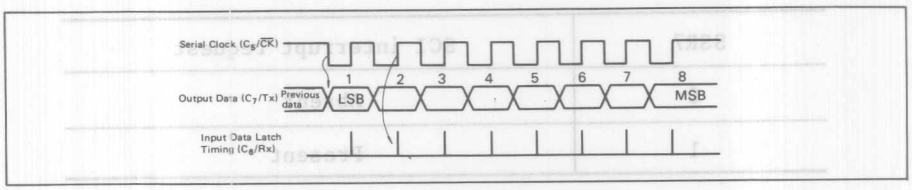


Fig. 2-9 SCI Timing Chart

(5) Data Reception

By writing the desired control bits into the SCI control register, a transfer rate and a source of transfer clock are determined and bit 6 and 5 of port C are set at the serial data input terminal and the serial clock terminal, respectively. Then dummy-writing or reading the SCI data register, the system is ready for receiving data. (This procedure is not needed after reading a subsequent received data. It must be taken after reset and after not reading a subsequent received data.)

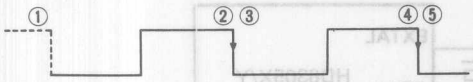
The data from the C_6/Rx terminal is input to the SCI data register synchronously with the leading edge of the serial clock (Refer to Fig. 2-9). When 8 bits of data have been received, the interrupt request bit is set in bit 7 of the SCI status register. This request can be masked by setting bit 5 of the SCI status register. If an external clock source has been selected, the transfer rate determined by bits 0 ∿ 3 of the SCI control register is ignored and the data is received synchronously with the clock from the $C_5/\overline{CK}$ terminal. If the internal clock has been selected, the $C_5/\overline{CK}$ terminal is set as output and clocks are output at the transfer rate selected by bits 0 ∿ 3 of the SCI control register.

(6) $TIMER_2$

The SCI transfer clock generator can be used as a timer. The SCI clock selected by bits 3 ∿ 0 of the SCI Control Register (4 μs ∿ approx. 32ms in 4MHz operation) is received by bit 6 of the SCI Status Register, and the timer 2 interrupt request bit is set at each falling edge of the SCI clock. Since this interrupt occurs periodically, $Timer_2$ is available for a reload counter or a timer.

$Timer_2$ is multiplexed with the SCI transfer clock generator. When using $Timer_2$ independently of the SCI, external clock should be selected as SCI clock source by setting both SCR5 and SCR4 to "1".

If Internal clock is selected as a SCI clock source, reading from or writing to the SDR initializes the prescalor of the SCI transfer clock generator.



① : Transfer clock generator is reset and mask bit (bit 4 of SCI status registor) is cleared

②,④ : $TIMER_2$ interrupt request

③,⑤ : $TIMER_2$ interrupt request bit cleared

2.5 Reset

The MCU can be reset either by external reset input ($\overline{RES}$) or power-on reset. (Refer to Fig. 2-10.) On power up, the reset input must be held "Low" for at least t_{osc} to assure that the internal oscillator is stabilized. A sufficient time of delay can be obtained by connecting a capacitance to the $\overline{RES}$ inputs as shown in Fig. 2-11.

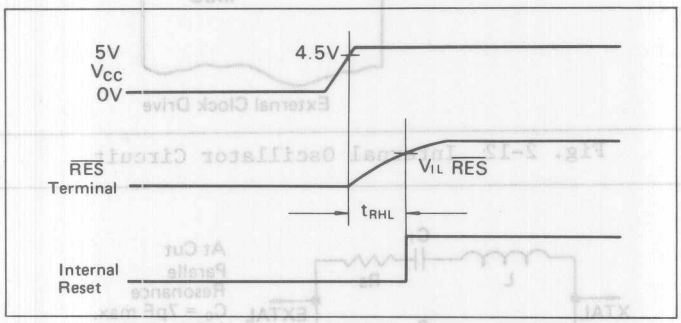


Fig. 2-10 Power On and Reset Timing

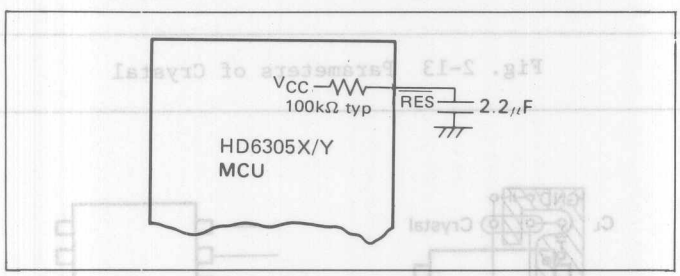


Fig. 2-11 Input Reset Delay Circuit

2.6 Internal Oscillator Options

The internal oscillator circuit is designed to meet the requirement for minimum external configurations. It can be driven by connecting a crystal (AT cut 2.0 ∿ 8.0MHz) or ceramic oscillator between pins 5 and 6 depending on the required oscillation frequency stability.

Three different terminal connections are shown in Fig. 2-12. Figs. 2-13 and 2-14 illustrate the specifications and typical arrangement of the crystal, respectively.

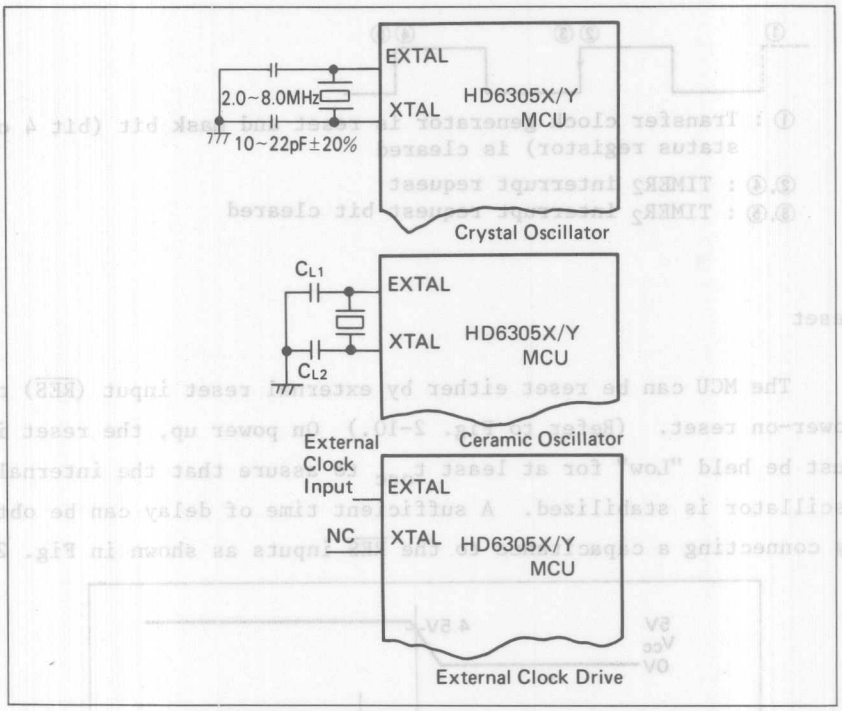


Fig. 2-12 Internal Oscillator Circuit

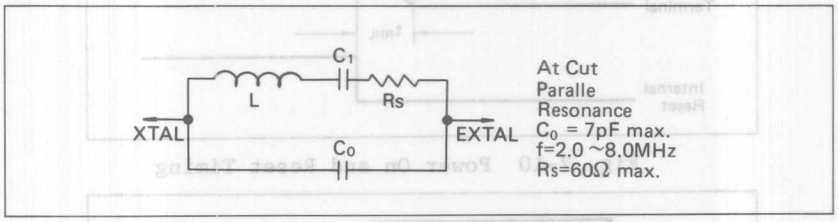
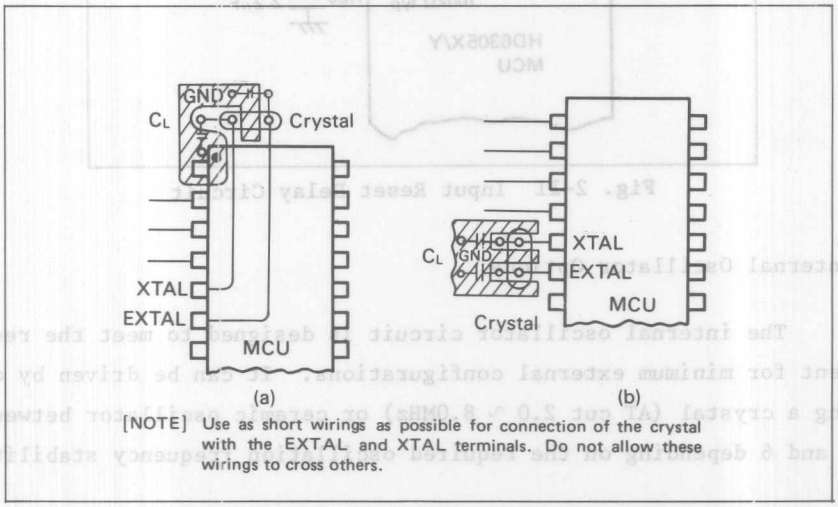


Fig. 2-13 Parameters of Crystal



[NOTE] Use as short wirings as possible for connection of the crystal with the EXTAL and XTAL terminals. Do not allow these wirings to cross others.

Fig. 2-14 Typical Crystal Arrangement

2.7 Interrupts

There are six interrupts: external interrupts ($\overline{INT}$, $\overline{INT_2}$), internal timer interrupts (TIMER, $TIMER_2$), serial interrupt (SCI) and interrupt by an instruction (SWI).

Of these six interrupts, the $\overline{INT_2}$ and TIMER interrupt or the SCI and $TIMER_2$ interrupt generate the same vector address, respectively.

When an interrupt occurs, the program in progress stops and the then CPU status is saved onto the stack. Then, the interrupt mask bit (I) of the condition code register is set and the start address of the interrupt processing routine is obtained from a particular interrupt vector address. Then the interrupt routine starts from the start address. System can exit from the interrupt routine by an RTI instruction. When this instruction is executed, the CPU status before the interrupt (saved onto the stack) is pulled and the CPU restarts the sequence with the instruction next to the one at which the interrupt occurred. Table 2-4 lists the priority of interrupts and their vector addresses.

A flowchart of the interrupt sequence is shown in Fig. 2-15. A block diagram of the interrupt request source is shown in Fig. 2-16.

(1) External Interrupt

In the block diagram, both the external interrupts $\overline{INT}$ and $\overline{INT_2}$ are edge trigger inputs. At the falling edge of each input, an interrupt request is generated and latched. The $\overline{INT}$ interrupt request is automatically cleared if jumping is made to the $\overline{INT}$ processing routine. The $\overline{INT_2}$ interrupt request is masked when "0" is written to bit 7 of the Miscellaneous Register.

If the I bit of the Condition Code Register is set to "1", interrupt requests of the external interrupt ($\overline{INT}$, $\overline{INT_2}$) are retained, but not processed. Immediately after the I bit is cleared, the CPU jumps to the corresponding interrupt routine. The $\overline{INT_2}$ interrupt can be masked by setting bit 6 of the Miscellaneous Register.

The $\overline{INT}$ terminal status can be tested by a BIL or BIH instruction. The $\overline{INT}$ falling edge detector circuit and its latching

circuit, and $\overline{INT_2}$ terminal are unaffected by executing BIL and BIH instructions.

(2) Internal Interrupt and SCI Interrupt

When I bit of the Condition Code Register is set to "1", internal timer interrupts (TIMER, $TIMER_2$) and SCI interrupt requests are retained without being processed. Immediately after I bit is cleared, the CPU initiates the corresponding interrupt service routine. Timer interrupt, SCI interrupt, $Timer_2$ interrupt can be masked by setting bit 6 of Timer Control Register, bit 5 of SCI Status Register, and bit 4 of SCI Status Register, respectively.

Table 2-4 Priority of Interrupts

Interrupt	Priority	Vector Address
$\overline{RES}$	1	$1FFE, $1FFF
SWI	2	$1FFC, $1FFD
$\overline{INT}$	3	$1FFA, $1FFB
TIMER/$\overline{INT_2}$	4	$1FF8, $1FF9
SCI/$TIMER_2$	5	$1FF6, $1FF7

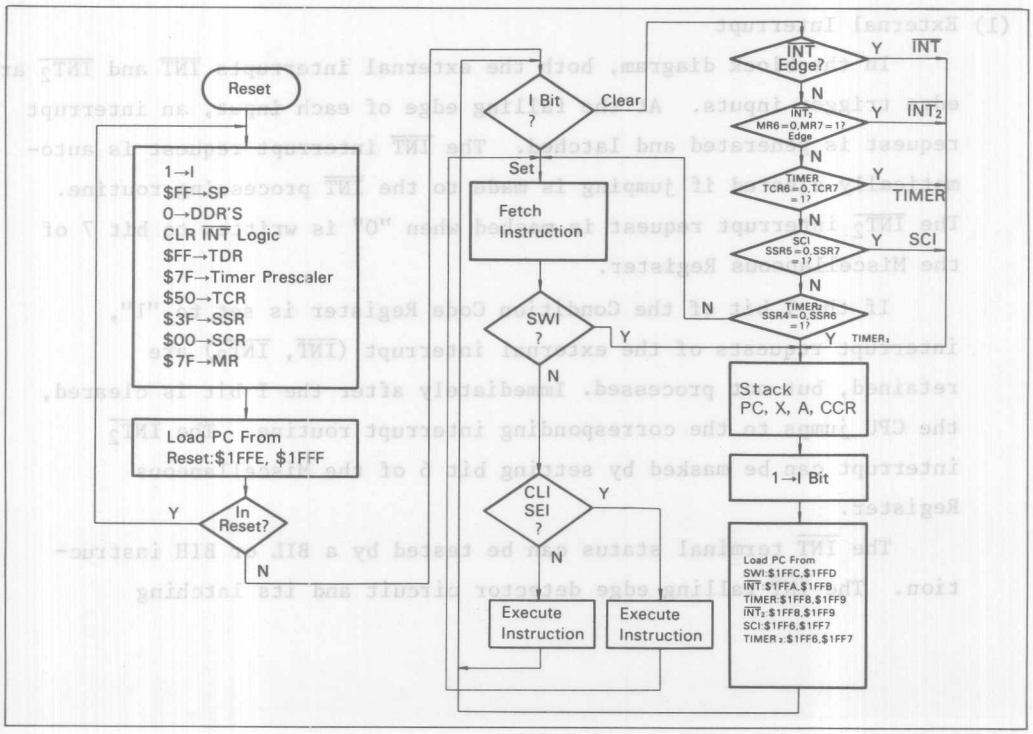


Fig. 2-15 Interrupt Flowchart

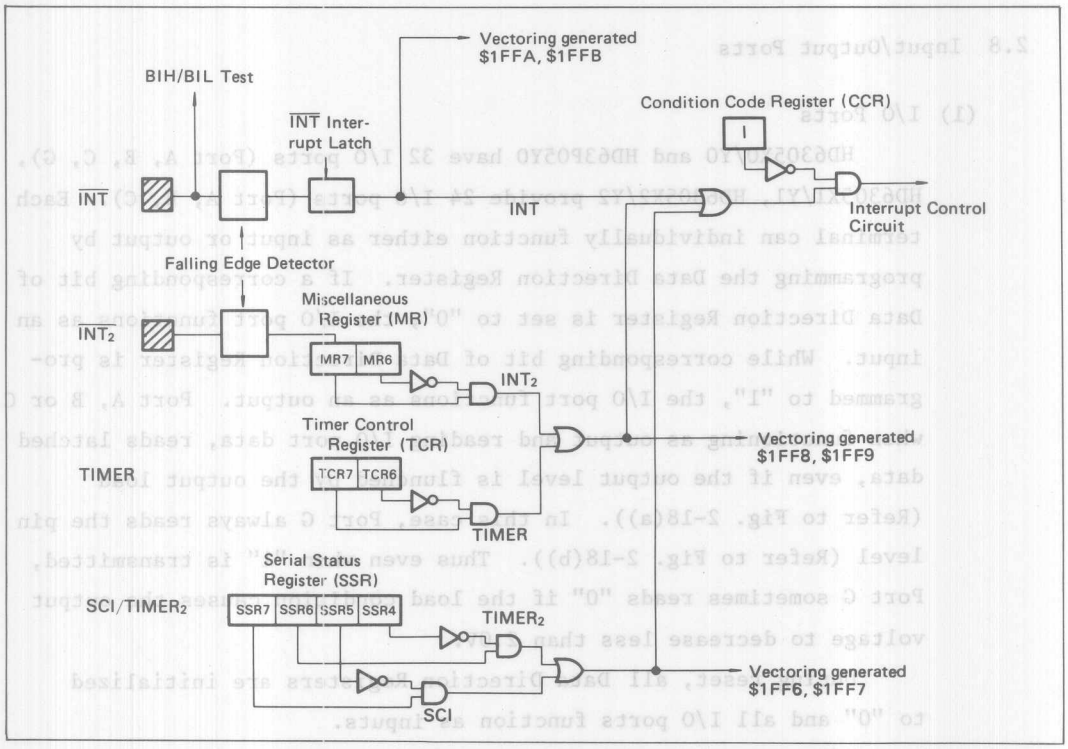


Fig. 2-16 Interrupt Request Generation Circuitry

(3) Miscellaneous Register (MR: $000A)

Miscellaneous Register (MR: $000A) specifies $\overline{INT}$ request sense (edge sense or level sense) and controls $\overline{INT_2}$ interrupt.

Bit 7 of Miscellaneous Register (MR7) is an $\overline{INT_2}$ interrupt request flag. When the falling edge is detected in $\overline{INT_2}$, MR7 is set to "1". MR7 can be checked by software if it is $\overline{INT_2}$ interrupt or not in the interrupt service routine (vector address: $1FF8, $1FF9). This bit can be reset by software.

Bit 6 (MR6) is the $\overline{INT_2}$ interrupt mask bit. If this bit is set to "1", the $\overline{INT_2}$ interrupt is masked. Both read and write are possible with bit 7, but "1" cannot be written in this bit by software. Thus an interrupt cannot be requested by software.

During reset, bit 7 is initialized to "0", while bit 6 is initialized to "1".

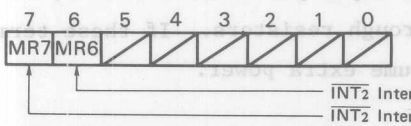


Fig. 2-17 Miscellaneous Register (MR:$000A)

2.8 Input/Output Ports

(1) I/O Ports

HD6305X0/Y0 and HD63P05Y0 have 32 I/O ports (Port A, B, C, G), HD6305X1/Y1, HD6305X2/Y2 provide 24 I/O ports (Port A, B, C). Each terminal can individually function either as input or output by programming the Data Direction Register. If a corresponding bit of Data Direction Register is set to "0", the I/O port functions as an input. While corresponding bit of Data Direction Register is programmed to "1", the I/O port functions as an output. Port A, B or C, when functioning as output and reading I/O port data, reads latched data, even if the output level is flunched by the output load (Refer to Fig. 2-18(a)). In this case, Port G always reads the pin level (Refer to Fig. 2-18(b)). Thus even when "1" is transmitted, Port G sometimes reads "0" if the load condition causes the output voltage to decrease less than 2.0V.

During reset, all Data Direction Registers are initialized to "0" and all I/O ports function as inputs.

I/O ports are compatible with TTL and CMOS in respect to both input and output.

If I/O ports are not used, they should be connected to Vss through resistors. If these terminals are left open, they may consume extra power.

(2) Output Ports (for HD6305X0/Y0, HD63P05Y0 only)

There are 16 output-only terminals (ports E and F). Each of them can also read. In this case, latched data is read even with the output terminal level being fluctuated by the output load (as with ports A, B and C).

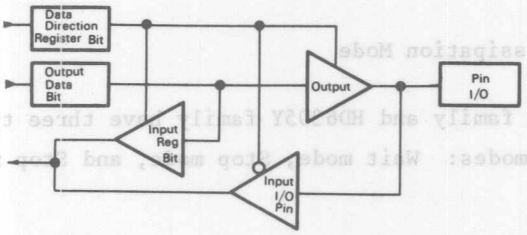
When reset, "Low" level is transmitted from each output terminal.

(3) Input Ports

Seven input-only terminals are available (port D). Writing to an input terminal is invalid.

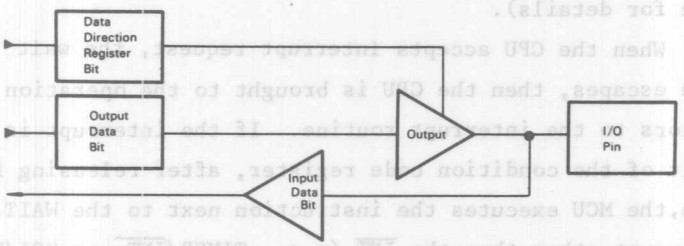
All input/output terminals, output terminals and input terminals are TLL compatible and CMOS compatible in respect of both input and output.

If I/O ports or input ports are not used, they should be connected to Vss through resistors. If these terminals are left open, they may consume extra power.



Bit of data direction register	Bit of output data	Status of output	Input to MCU
1	0	0	0
1	1	1	1
0	×	3-state	Pin

a. Ports A, B and C



b. Port G

Fig. 2-18 Input/Output Port Diagram

2.9 Low Power Dissipation Mode

HD6305X family and HD6305Y family have three types of low power dissipation modes: Wait mode, Stop mode, and Stop mode.

(1) Wait Mode

When Wait instruction is executed, the MCU enters into the WAIT mode; the oscillator does not stop, but the internal clock stops. The CPU stops but the internal peripherals (e.g. Timer and SCI) continue their current operations. (Note: Once the system entered into the Wait mode, SCI can no longer be retriggered.) During the Wait mode, RAM and I/O terminals hold their conditions just before entering into the Wait mode, while I bit of the Condition Code Register is cleared to "0".

The MCU can be recovered from the Wait Mode by an interrupt ($\overline{INT}$, TIMER/$\overline{INT_2}$, or SCI/$TIMER_2$), $\overline{RES}$ or $\overline{STBY}$. The $\overline{RES}$ resets the MCU, and the $\overline{STBY}$ brings it into the standby mode (Refer to Standby mode for details).

When the CPU accepts interrupt request, the wait mode escapes, then the CPU is brought to the operation mode and vectors to the interrupt routine. If the interrupt is masked by the I bit of the condition code register, after releasing from the wait mode, the MCU executes the instruction next to the WAIT. If an interrupt other than the $\overline{INT}$ (e.g., TIMER/$\overline{INT_2}$ or SCI/$TIMER_2$) is masked by the timer control register, miscellaneous register or serial status register, the CPU does not receive any interrupt request, Thus the wait mode cannot be released.

Fig. 2-19 shows a flowchart for the wait function.

(2) Stop Mode

When STOP instruction is being executed, the MCU enters into the stop mode. In this mode, the oscillator stops and the CPU and peripheral stops functioning but the RAM, registers and I/O terminals hold their condition just before entering into the stop mode.

The escape from this mode can be done by an external interrupt ($\overline{INT}$ or $\overline{INT_2}$), $\overline{RES}$ or $\overline{STBY}$. The $\overline{RES}$ resets the MCU and the $\overline{STBY}$ brings it into the standby mode.

When the CPU accepts interrupt request, the stop mode escapes, then the CPU is brought to the operation mode and vectors to the interrupt routine. If the interrupt is masked by the I bit of the condition code register, after releasing from the stop mode, the MCU executes the instruction next to the STOP. The MCU cannot be released from the STOP mode if the $\overline{INT_2}$ interrupt is masked by the miscellaneous register.

Fig. 2-20 shows a flowchart for the stop function. Fig. 2-21 shows a timing chart when the MCU is released from the stop mode.

For releasing from the stop mode by an interrupt, oscillation starts upon input of the interrupt and, after the internal delay time for stabilized oscillation, the CPU becomes active. When restarting by $\overline{RES}$, oscillation starts when the $\overline{RES}$ goes "0" and the CPU restarts when the $\overline{RES}$ goes "1". The duration of $\overline{RES}$="0" must exceed t_{osc}=20ms to assure stabilized oscillation.

(3) Standby Mode

The MCU enters into the standby mode when the $\overline{STBY}$ goes "Low". In this mode, all operations stop and the internal condition is reset holding the RAM contents. The I/O terminals go into High-impedance state. The standby mode should escape by bringing $\overline{STBY}$ "High". The CPU must be restarted by reset. The timing of input signals at the $\overline{RES}$ and $\overline{STBY}$ is shown in Fig. 2-22.

Table 2-5 lists the status of each parts of the MCU in each low power dissipation modes. Transitions between each mode are shown in Fig. 2-23.

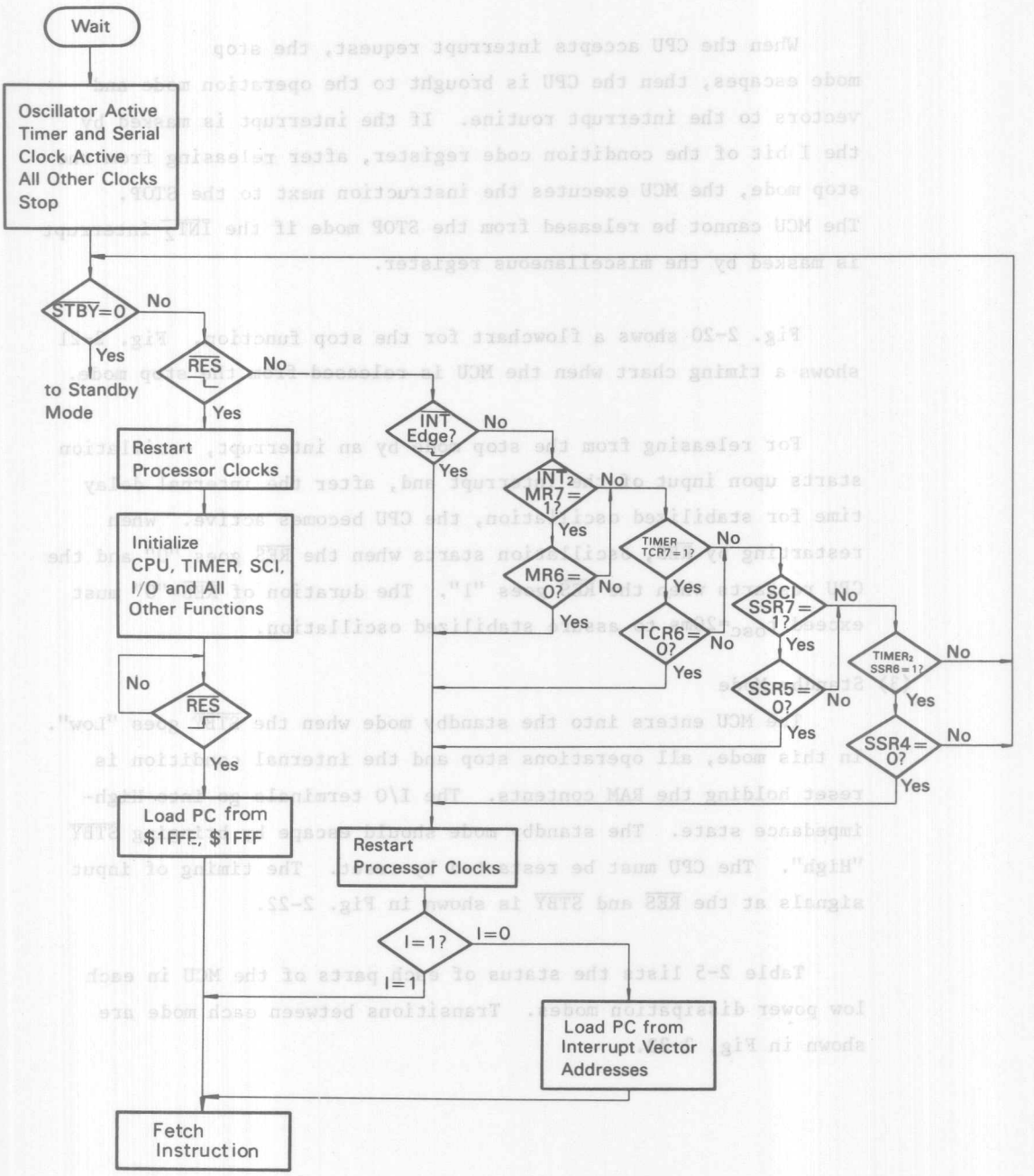


Fig. 2-19 Wait Mode Flowchart

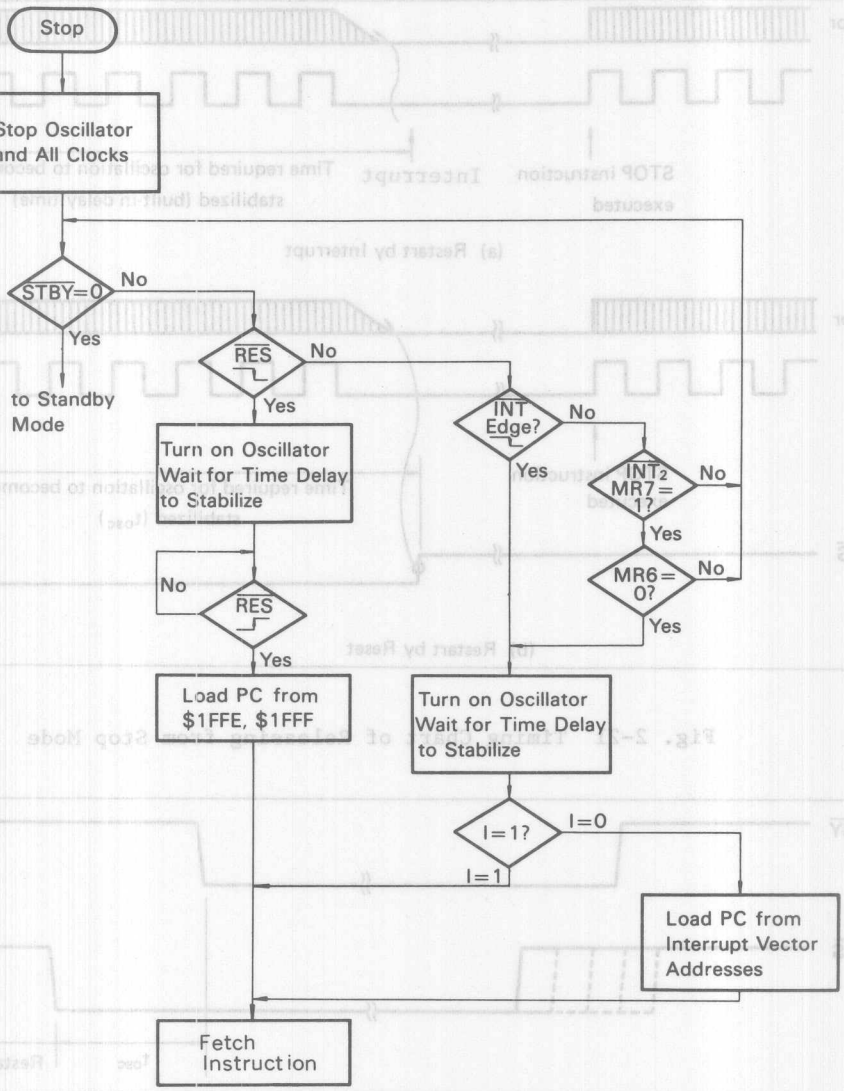


Fig. 2-20 Stop Mode Flowchart

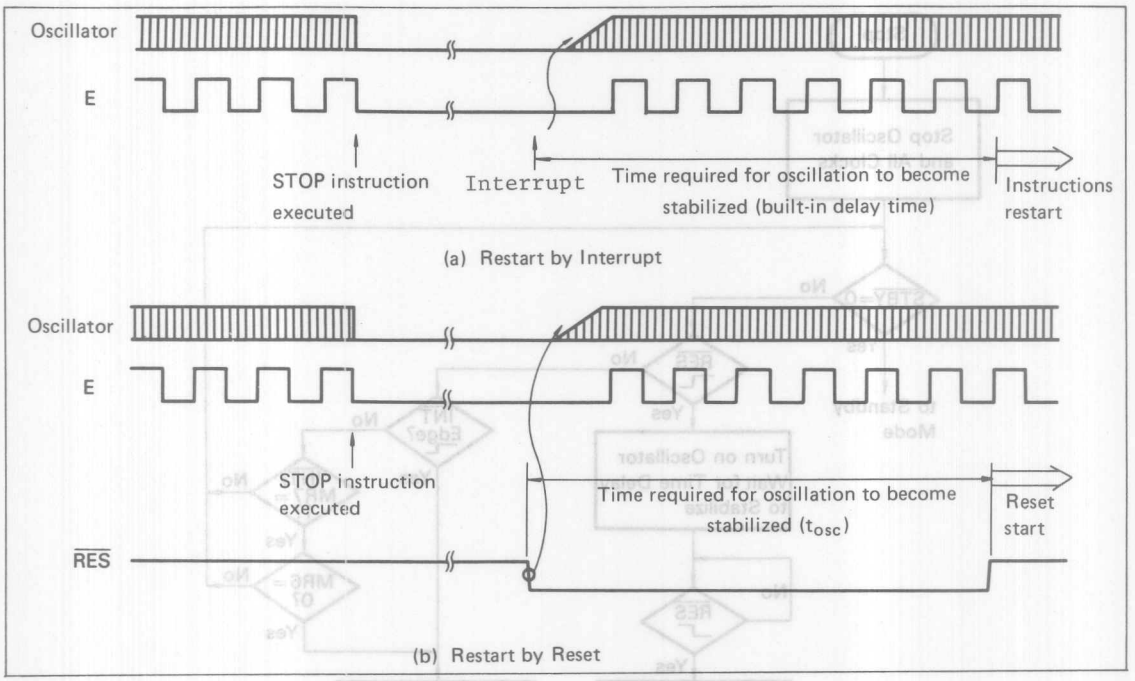


Fig. 2-21 Timing Chart of Releasing from Stop Mode

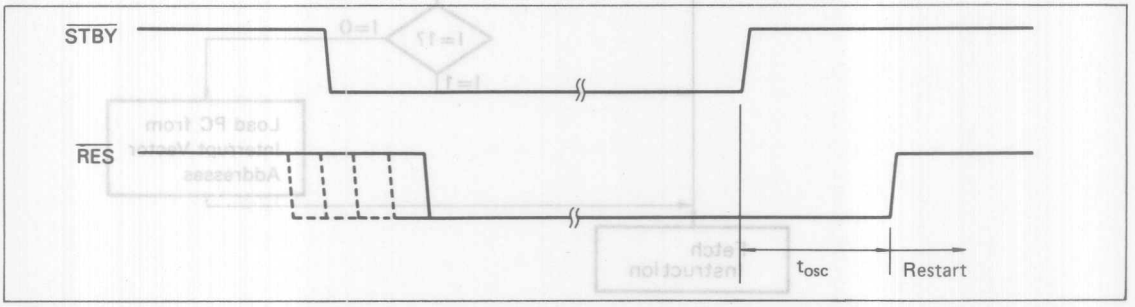


Fig. 2-22 Timing Chart of Releasing from Standby Mode

Table 2-5 Status of Each Part of MCU in Low Power Dissipation Modes

Mode	Start		Condition						Escape
			Oscillator	CPU	Timer, Serial	Register	RAM	I/O terminal	
WAIT	Software	WAIT instruction	Active	Stop	Active	Keep	Keep	Keep	$\overline{STBY}$, $\overline{RES}$, $\overline{INT}$, $\overline{INT_2}$, each interrupt request of TIMER, $TIMER_2$, SCI
STOP		STOP instruction	Stop	Stop	Stop	Keep	Keep	Keep	$\overline{STBY}$, $\overline{RES}$, $\overline{INT}$, $\overline{INT_2}$
Standby	Hardware	$\overline{STBY}$="Low"	Stop	Stop	Stop	Reset	Keep	High impedance	$\overline{STBY}$="High"

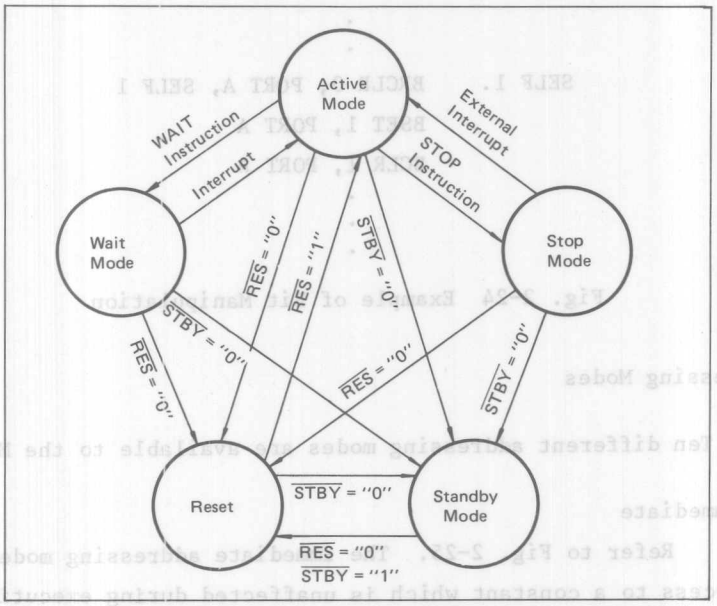


Fig. 2-23 Transitions among Active Mode, Wait Mode, Stop Mode, Standby Mode and Reset

2.10 Bit Manipulation

The HD6305X and HD6305Y family sets or clears 1-bit of the RAM or an I/O port (except the write-only registers such as the data direction register) with a single instruction, (BSET or BCLR).

Every bit of memory or I/O within page 0 ($00 ∿ $FF) can be tested by the BRSET or BRCLR instruction. The program can branch to required destinations depending on the result of the test. Since bits in the RAM, or I/O can be manipulated, the user may use a bit within the RAM as a flag or handle a single I/O bit as an independent I/O terminal. Fig. 2-24 shows an example of bit manipulation and the validity of test instructions. In the example, the program is configured assuming that bit 0 of port A is connected to a zero cross detector circuit and bit 1 of the same port to the trigger of a triac.

This program can activate the triac within 10μs from zero-crossing just by using 7 bytes on the ROM. The internal timer provides a required delay time and pulse width modulation of power as well.

```
        .
        .
        .
SELF 1. BRCLR 0, PORT A, SELF 1
        BSET 1, PORT A
        BCLR 1, PORT A
        .
        .
        .
```

Fig. 2-24 Example of Bit Manipulation

2.11 Addressing Modes

Ten different addressing modes are available to the MCU.

(1) Immediate

Refer to Fig. 2-25. The immediate addressing mode provides access to a constant which is unaffected during executing the program.

This instruction requires 2 bytes. The effective address (EA) is PC and the operand is fetched from the byte that follows the operation code.

(2) Direct

Refer to Fig. 2-26. In the direct addressing mode, the address of the operand is contained in the 2nd byte of the instruction. The user can gain direct access to memory up to the lower 255th address. All RAM and I/O registers are on page 0 of address space so that the direct addressing mode may be utilized.

(3) Extended

Refer to Fig. 2-27. The extended addressing is used for referring to all addresses of memory. The EA is the contents of the 2 bytes that follow the operation code. An extended addressing instruction requires 3 bytes long.

(4) Relative

Refer to Fig. 2-28. The relative addressing mode is used with only branch instructions. When a branch occurs, the program counter is loaded with the contents of the byte following the operation code. EA = (PC) + 2 + Rel., where Rel. indicates a signed 8-bit data following the operation code. If no branch occurs, Rel. = 0. When a branch occurs, the program jumps to any byte in the range +129 to -127. A branch instruction requires 2

bytes.

(5) Indexed (No Offset)

Refer to Fig. 2-29. The indexed addressing mode allows the MCU to be accessed up to the lower 255th address of memory. In this mode, an instruction requires one byte. The EA is the contents of the index register.

(6) Indexed (8-bit Offset)

Refer to Fig. 2-30. The EA is the contents of the byte following to the operation code, plus the contents of the index register. This mode allows access up to the lower 511th address of memory. Each instruction, when used in the index addressing mode, (8-bit offset) requires 2 bytes.

(7) Indexed (16-bit Offset)

Refer to Fig. 2-31. The contents of the 2 bytes following to the operation code are added to content of the index register to compute the value of EA. In this mode, the complete memory can be accessed. When used in the indexed addressing mode (16-bit offset), an instruction requires 3 bytes.

(8) Bit Set/Clear

Refer to Fig. 2-32. This addressing mode is applied to the BSET and BCLR instructions that can set or clear any bit on page 0. The lower 3 bits of the operation code specify the bit to be set or cleared. The byte that follows the operation code indicates an address within page 0.

(9) Bit Test and Branch

Refer to Fig. 2-33. This addressing mode is applied to the BRSET and BRCLR instructions that can test any bit within page 0 and can be branched in the relative addressing mode. The byte to be tested is addressed depending on the contents of the byte following the operation code. Individual bits within the byte to be tested are specified by the lower 3 bits of the operation code. The 3rd byte represents a relative value which will be added to the program counter when a branch condition is established. Each of these instructions requires 3 bytes. The value of the test bit is written in the carry bit of the condition code register.

(10) Implied

Refer to Fig. 2-34. This mode contains no EA. All informa-

tion necessary for execution of an instruction is contained in the operation code. Direct manipulation on the accumulator and index register is included in the implied addressing mode. Other instructions such as SWI and RTI are also used in this mode. All instructions used in the implied addressing mode requires one byte.

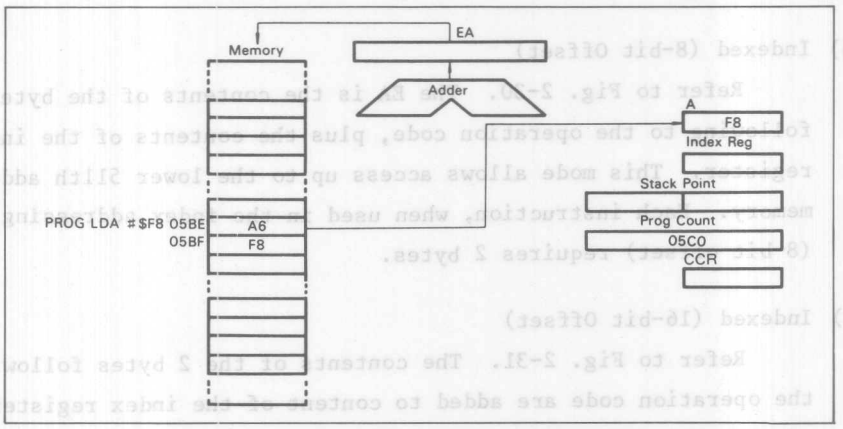


Fig. 2-25 Example of Immediate Addressing

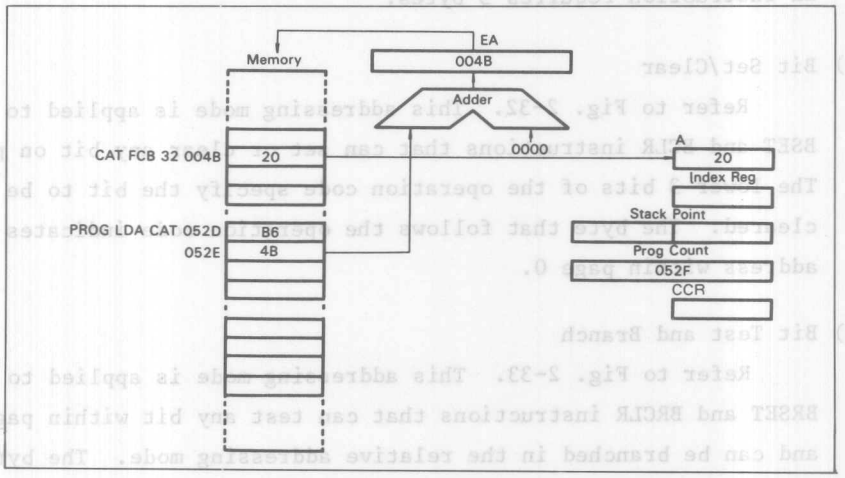


Fig. 2-26 Example of Direct Addressing

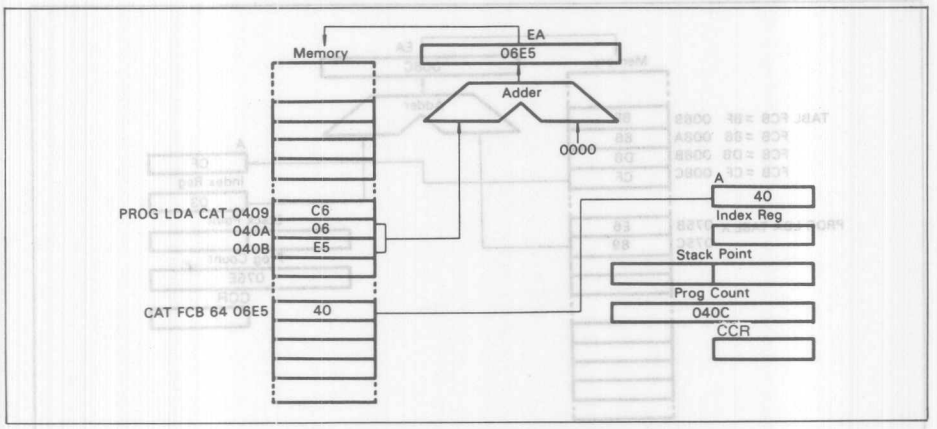


Fig. 2-27 Example of Extended Addressing

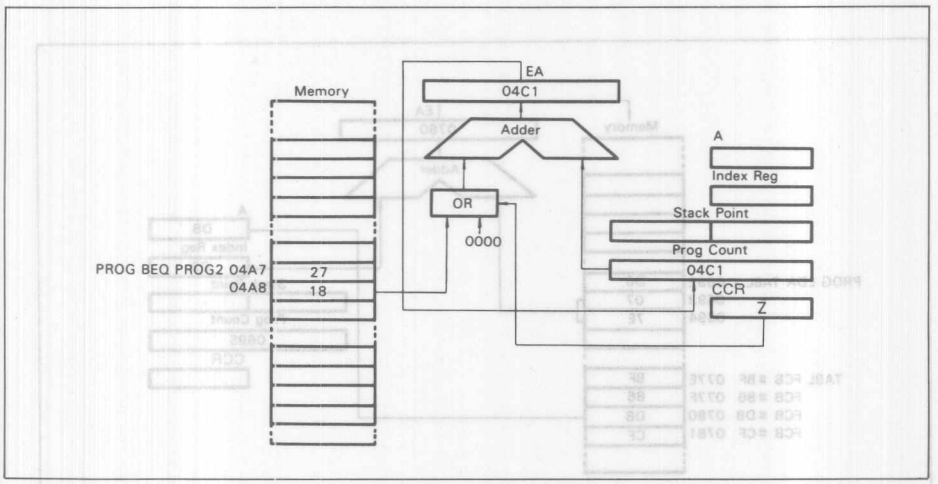


Fig. 2-28 Example of Relative Addressing

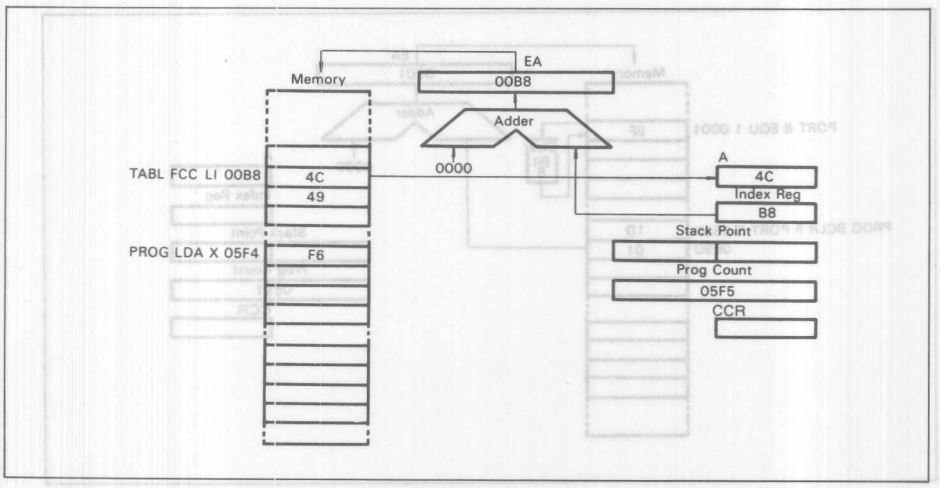


Fig. 2-29 Example of Indexed (No Offset) Addressing

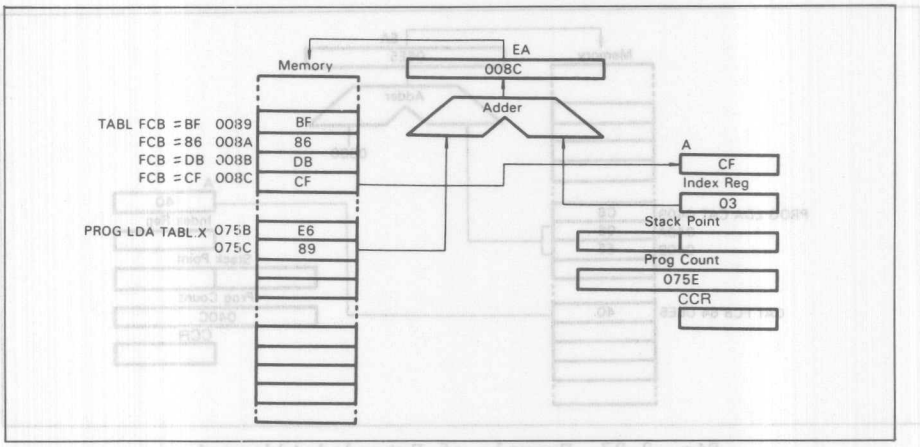


Fig. 2-30 Example of Index (8-bit Offset) Addressing

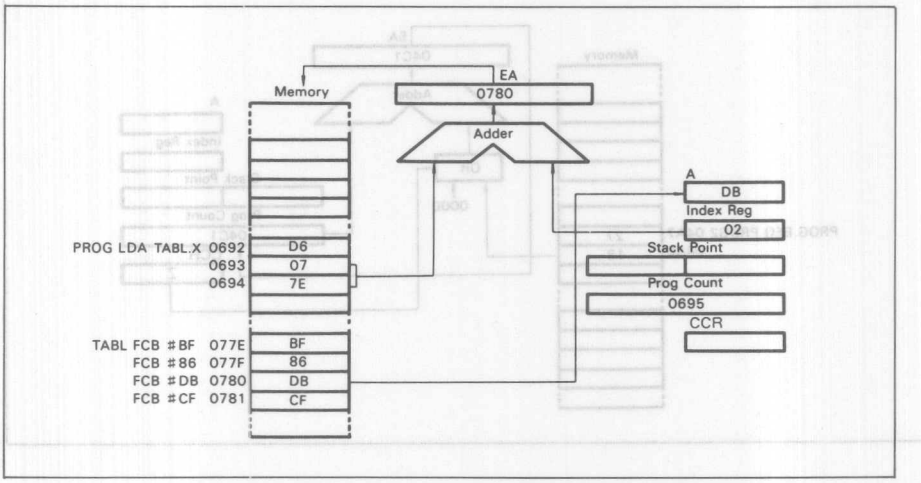


Fig. 2-31 Example of Index (16-bit Offset) Addressing

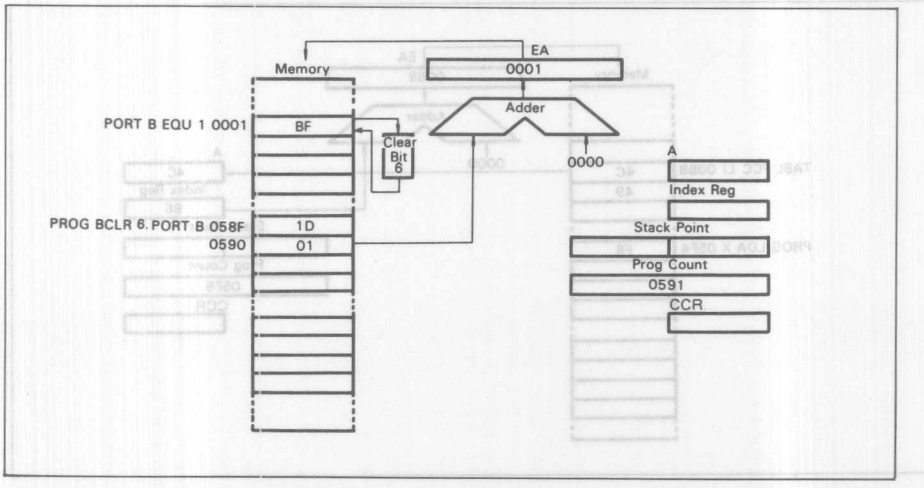


Fig. 2-32 Example of Bit Set/Clear Addressing

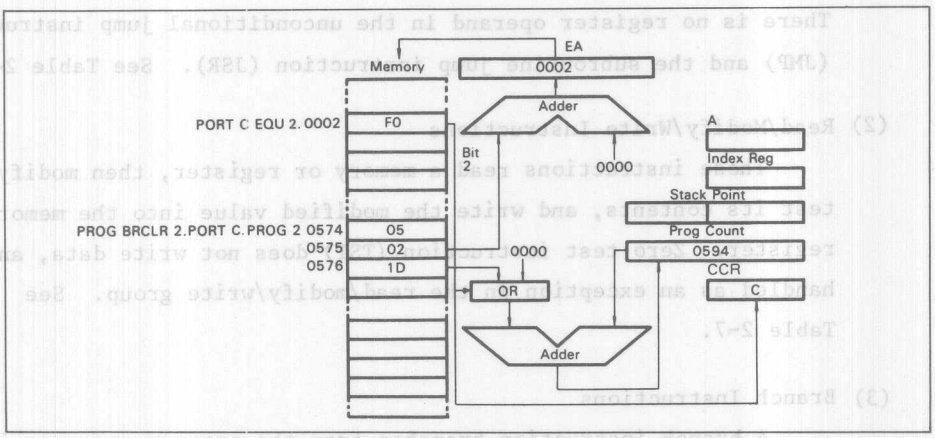


Fig. 2-33 Example of Bit Test and Branch Addressing

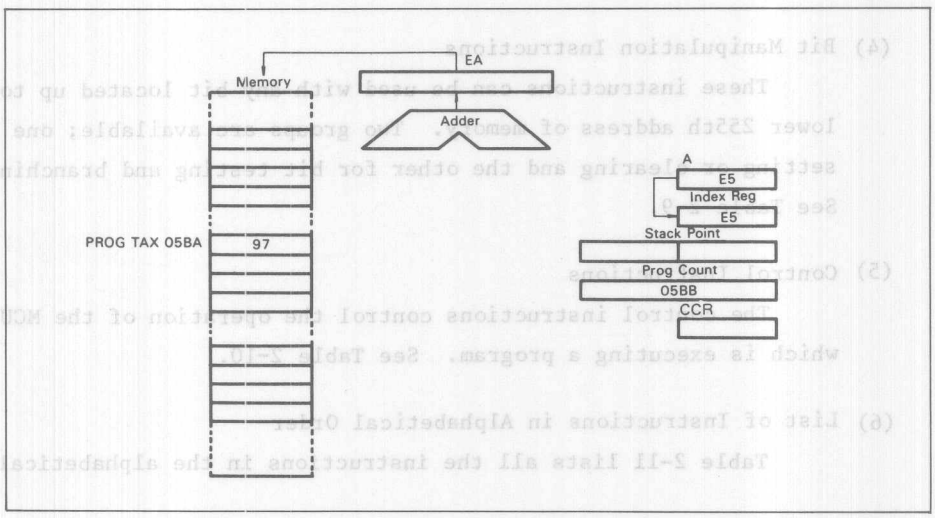


Fig. 2-34 Example of Implied Addressing

2.12 Instruction Set

62 basic instructions are available to the HD6305X and the HD6305Y MCU. They can be classified into five categories: register/memory, read/modify/write, branch, bit manipulation, and control. The details of each instruction are described in Tables 2-6 through 2-12.

(1) Register/Memory Instructions

Most of these instructions use two operands. One operand is either an accumulator or index register. The other is derived from memory using one of the addressing modes used on the MCU.

There is no register operand in the unconditional jump instruction (JMP) and the subroutine jump instruction (JSR). See Table 2-6.

(2) Read/Modify/Write Instructions

These instructions read a memory or register, then modify or test its contents, and write the modified value into the memory or register. Zero test instruction (TST) does not write data, and is handled as an exception in the read/modify/write group. See Table 2-7.

(3) Branch Instructions

A branch instruction branches from the program sequence in progress if a particular condition is established. See Table 2-8.

(4) Bit Manipulation Instructions

These instructions can be used with any bit located up to the lower 255th address of memory. Two groups are available; one for setting or clearing and the other for bit testing and branching. See Table 2-9.

(5) Control Instructions

The control instructions control the operation of the MCU which is executing a program. See Table 2-10.

(6) List of Instructions in Alphabetical Order

Table 2-11 lists all the instructions in the alphabetical order.

(7) Operation Code Map

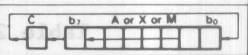
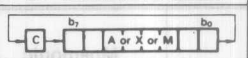
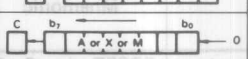
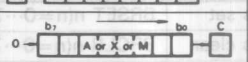
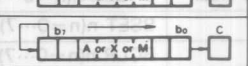
Table 2-12 shows the operation code map for the instructions used on the MCU.

Table 2-6 Register/Memory Instructions

Operations	Mnemonic	Addressing Modes: Immediate OP	#	~	Direct OP	#	~	Extended OP	#	~	Indexed (No Offset) OP	#	~	Indexed (8-Bit Offset) OP	#	~	Indexed (16-Bit Offset) OP	#	~	Boolean/ Arithmetic Operation	Condition Code H	I	N	Z	C
Load A from Memory	LDA	A6	2	2	B6	2	3	C6	3	4	F6	1	3	E6	2	4	D6	3	5	M→A	•	•	^	^	•
Load X from Memory	LDX	AE	2	2	BE	2	3	CE	3	4	FE	1	3	EE	2	4	DE	3	5	M→X	•	•	^	^	•
Store A in Memory	STA	–	–	–	B7	2	3	C7	3	4	F7	1	4	E7	2	4	D7	3	5	A→M	•	•	^	^	•
Store X in Memory	STX	–	–	–	BF	2	3	CF	3	4	FF	1	4	EF	2	4	DF	3	5	X→M	•	•	^	^	•
Add Memory to A	ADD	AB	2	2	BB	2	3	CB	3	4	FB	1	3	EB	2	4	DB	3	5	A+M→A	^	•	^	^	^
Add Memory and Carry to A	ADC	A9	2	2	B9	2	3	C9	3	4	F9	1	3	E9	2	4	D9	3	5	A+M+C→A	^	•	^	^	^
Subtract Memory	SUB	A0	2	2	B0	2	3	C0	3	4	F0	1	3	E0	2	4	D0	3	5	A−M→A	•	•	^	^	^
Subtract Memory from A with Borrow	SBC	A2	2	2	B2	2	3	C2	3	4	F2	1	3	E2	2	4	D2	3	5	A−M−C→A	•	•	^	^	^
AND Memory to A	AND	A4	2	2	B4	2	3	C4	3	4	F4	1	3	E4	2	4	D4	3	5	A · M→A	•	•	^	^	•
OR Memory with A	ORA	AA	2	2	BA	2	3	CA	3	4	FA	1	3	EA	2	4	DA	3	5	A+M→A	•	•	^	^	•
Exclusive OR Memory with A	EOR	A8	2	2	B8	2	3	C8	3	4	F8	1	3	E8	2	4	D8	3	5	A⊕M→A	•	•	^	^	•
Arithmetic Compare A with Memory	CMP	A1	2	2	B1	2	3	C1	3	4	F1	1	3	E1	2	4	D1	3	5	A−M	•	•	^	^	^
Arithmetic Compare X with Memory	CPX	A3	2	2	B3	2	3	C3	3	4	F3	1	3	E3	2	4	D3	3	5	X−M	•	•	^	^	^
Bit Test Memory with A (Logical Compare)	BIT	A5	2	2	B5	2	3	C5	3	4	F5	1	3	E5	2	4	D5	3	5	A · M	•	•	^	^	•
Jump Unconditional	JMP				BC	2	2	CC	3	3	FC	1	2	EC	2	3	DC	3	4		•	•	•	•	•
Jump to Subroutine	JSR				BD	2	5	CD	3	6	FD	1	5	ED	2	5	DD	3	6		•	•	•	•	•

Symbols: Op = Operation
= Number of bytes
~ = Number of cycles

Table 2-7 Read/Modify/Write Instructions

Operations	Mnemonic	Addressing Modes: Implied(A) OP	#	~	Implied(X) OP	#	~	Direct OP	#	~	Indexed (No Offset) OP	#	~	Indexed (8-Bit Offset) OP	#	~	Boolean/Arithmetic Operation	Condition Code H	I	N	Z	C
Increment	INC	4C	1	2	5C	1	2	3C	2	5	7C	1	5	6C	2	6	A+1→A or X+1→X or M+1→M	•	•	^	^	•
Decrement	DEC	4A	1	2	5A	1	2	3A	2	5	7A	1	5	6A	2	6	A−1→A or X−1→X or M−1→M	•	•	^	^	•
Clear	CLR	4F	1	2	5F	1	2	3F	2	5	7F	1	5	6F	2	6	00→A or 00→X or 00→M	•	•	0	1	•
Complement	COM	43	1	2	53	1	2	33	2	5	73	1	5	63	2	6	$\overline{A}$→A or $\overline{X}$→X or $\overline{M}$→M	•	•	^	^	1
Negate (2's Complement)	NEG	40	1	2	50	1	2	30	2	5	70	1	5	60	2	6	00−A→A or 00−X→X or 00−M→M	•	•	^	^	^
Rotate Left Thru Carry	ROL	49	1	2	59	1	2	39	2	5	79	1	5	69	2	6		•	•	^	^	^
Rotate Right Thru Carry	ROR	46	1	2	56	1	2	36	2	5	76	1	5	66	2	6		•	•	^	^	^
Logical Shift Left	LSL	48	1	2	58	1	2	38	2	5	78	1	5	68	2	6		•	•	^	^	^
Logical Shift Right	LSR	44	1	2	54	1	2	34	2	5	74	1	5	64	2	6		•	•	0	^	^
Arithmetic Shift Right	ASR	47	1	2	57	1	2	37	2	5	77	1	5	67	2	6		•	•	^	^	^
Arithmetic Shift Left	ASL	48	1	2	58	1	2	38	2	5	78	1	5	68	2	6	Equal to LSL	•	•	^	^	^
Test for Negative or Zero	TST	4D	1	2	5D	1	2	3D	2	4	7D	1	4	6D	2	5	A−00 or X−00 or M−00	•	•	^	^	•

Symbols: Op = Operation
= Number of bytes
~ = Number of cycles

Table 2-8 Branch Instructions

Operations	Mnemonic	Addressing Modes			Branch Test	Condition Code				
		Relative								
		OP	#	~		H	I	N	Z	C
Branch Always	BRA	20	2	3	None	●	●	●	●	●
Branch Never	BRN	21	2	3	None	●	●	●	●	●
Branch IF Higher	BHI	22	2	3	C+Z=0	●	●	●	●	●
Branch IF Lower or Same	BLS	23	2	3	C+Z=1	●	●	●	●	●
Branch IF Carry Clear	BCC	24	2	3	C=0	●	●	●	●	●
(Branch IF Higher or Same)	(BHS)	24	2	3	C=0	●	●	●	●	●
Branch IF Carry Set	BCS	25	2	3	C=1	●	●	●	●	●
(Branch IF Lower)	(BLO)	25	2	3	C=1	●	●	●	●	●
Branch IF Not Equal	BNE	26	2	3	Z=0	●	●	●	●	●
Branch IF Equal	BEQ	27	2	3	Z=1	●	●	●	●	●
Branch IF Half Carry Clear	BHCC	28	2	3	H=0	●	●	●	●	●
Branch IF Half Carry Set	BHCS	29	2	3	H=1	●	●	●	●	●
Branch IF Plus	BPL	2A	2	3	N=0	●	●	●	●	●
Branch IF Minus	BMI	2B	2	3	N=1	●	●	●	●	●
Branch IF Interrupt Mask Bit is Clear	BMC	2C	2	3	I=0	●	●	●	●	●
Branch IF Interrupt Mask Bit is Set	BMS	2D	2	3	I=1	●	●	●	●	●
Branch IF Interrupt Line is Low	BIL	2E	2	3	INT=0	●	●	●	●	●
Branch IF Interrupt Line is High	BIH	2F	2	3	INT=1	●	●	●	●	●
Branch to Subroutine	BSR	AD	2	5	——	●	●	●	●	●

Symbols: Op = Operation
= Number of bytes
~ = Number of cycles

Table 2-9 Bit Manipulation Instructions

Operations	Mnemonic	Addressing Modes						Boolean/ Arithmetic Operation	Branch Test	Condition Code				
		Bit Set/Clear			Bit Test and Branch									
		OP	#	~	OP	#	~			H	I	N	Z	C
Branch IF Bit n is set	BRSET n(n=0···7)	—	—	—	2·n	3	5	——	Mn=1	●	●	●	●	∧
Branch IF Bit n is clear	BRCLR n(n=0···7)	—	—	—	01+2·n	3	5	——	Mn=0	●	●	●	●	∧
Set Bit n	BSET n(n=0···7)	10+2·n	2	5	—	—	—	1→Mn	——	●	●	●	●	●
Clear Bit n	BCLR n(n=0···7)	11+2·n	2	5	—	—	—	0→Mn	——	●	●	●	●	●

Symbols: Op = Operation
= Number of bytes
~ = Number of cycles

Table 2-10 Control Instructions

Operations	Mnemonic	Addressing Modes: Implied OP	#	~	Boolean Operation	Condition Code: H	I	N	Z	C
Transfer A to X	TAX	97	1	2	A→X	●	●	●	●	●
Transfer X to A	TXA	9F	1	2	X→A	●	●	●	●	●
Set Carry Bit	SEC	99	1	1	1→C	●	●	●	●	1
Clear Carry Bit	CLC	98	1	1	0→C	●	●	●	●	0
Set Interrupt Mask Bit	SEI	9B	1	2	1→I	●	1	●	●	●
Clear Interrupt Mask Bit	CLI	9A	1	2	0→I	●	0	●	●	●
Software Interrupt	SWI	83	1	10		●	1	●	●	●
Return from Subroutine	RTS	81	1	5		●	●	●	●	●
Return from Interrupt	RTI	80	1	8		?	?	?	?	?
Reset Stack Pointer	RSP	9C	1	2	$FF→SP	●	●	●	●	●
No-Operation	NOP	9D	1	1	Advance Prog. Cntr. Only	●	●	●	●	●
Decimal Adjust A	DAA	8D	1	2	Converts binary add of BCD charcters into BCD format	●	●	∧	∧	∧*
Stop	STOP	8E	1	4		●	●	●	●	●
Wait	WAIT	8F	1	4		●	●	●	●	●

Symbols: Op = Operation
= Number of bytes
~ = Number of cycles

* Are BCD characters of upper byte 10 or more? (They are not cleared if set in advance.)

Table 2-11 Instruction Set (in Alphabetical Order)

Mnemonic	Addressing Modes: Implied	Immediate	Direct	Extended	Relative	Indexed (No Offset)	Indexed (8-Bit)	Indexed (16-Bit)	Bit Set/ Clear	Bit Test & Branch	Condition Code: H	I	N	Z	C
ADC		×	×	×		×	×	×			∧	●	∧	∧	∧
ADD		×	×	×		×	×	×			∧	●	∧	∧	∧
AND		×	×	×		×	×	×			●	●	∧	∧	●
ASL	×		×			×	×				●	●	∧	∧	∧
ASR	×		×			×	×				●	●	∧	∧	∧
BCC					×						●	●	●	●	●
BCLR									×		●	●	●	●	●
BCS					×						●	●	●	●	●
BEQ					×						●	●	●	●	●
BHCC					×						●	●	●	●	●
BHCS					×						●	●	●	●	●
BHI					×						●	●	●	●	●
(BHS)					×						●	●	●	●	●
BIH					×						●	●	●	●	●
BIL					×						●	●	●	●	●
BIT		×	×	×		×	×	×			●	●	∧	∧	●
(BLO)					×						●	●	●	●	●
BLS					×						●	●	●	●	●
BMC					×						●	●	●	●	●
BMI					×						●	●	●	●	●
BMS					×						●	●	●	●	●
BNE					×						●	●	●	●	●
BPL					×						●	●	●	●	●
BRA					×						●	●	●	●	●

(to be continued)

Condition Code Symbols:

H	Half Carry (From Bit 3)	C	Carry/Borrow
I	Interrupt Mask	∧	Test and Set if True, Cleared Otherwise
N	Negative (Sign Bit)	●	Not Affected
Z	Zero	?	Load CC Register From Stack

Mnemonic	Addressing Modes										Condition Code				
	Implied	Immediate	Direct	Extended	Relative	Indexed (No Offset)	Indexed (8-Bit)	Indexed (16-Bit)	Bit Set/ Clear	Bit Test & Branch	H	I	N	Z	C
BRN					×						●	●	●	●	●
BRCLR										×	●	●	●	●	∧
BRSET										×	●	●	●	●	∧
BSET									×		●	●	●	●	●
BSR					×						●	●	●	●	●
CLC	×										●	●	●	●	0
CLI	×										●	0	●	●	●
CLR	×		×			×	×				●	●	0	1	●
CMP		×	×	×		×	×	×			●	●	∧	∧	∧
COM	×		×			×	×				●	●	∧	∧	1
CPX		×	×	×		×	×	×			●	●	∧	∧	∧
DAA	×										●	●	∧	∧	∧
DEC	×		×			×	×				●	●	∧	∧	●
EOR		×	×	×		×	×	×			●	●	∧	∧	●
INC	×		×			×	×				●	●	∧	∧	●
JMP			×	×		×	×	×			●	●	●	●	●
JSR			×	×		×	×	×			●	●	●	●	●
LDA		×	×	×		×	×	×			●	●	∧	∧	●
LDX		×	×	×		×	×	×			●	●	∧	∧	●
LSL	×		×			×	×				●	●	∧	∧	∧
LSR	×		×			×	×				●	●	0	∧	∧
NEG	×		×			×	×				●	●	∧	∧	∧
NOP	×										●	●	●	●	●
ORA		×	×	×		×	×	×			●	●	∧	∧	●
ROL	×		×			×	×				●	●	∧	∧	∧
ROR	×		×			×	×				●	●	∧	∧	∧
RSP	×										●	●	●	●	●
RTI	×										?	?	?	?	?
RTS	×										●	●	●	●	●
SBC		×	×	×		×	×	×			●	●	∧	∧	∧
SEC	×										●	●	●	●	1
SEI	×										●	1	●	●	●
STA			×	×		×	×	×			●	●	∧	∧	●
STOP	×										●	●	●	●	●
STX			×	×		×	×	×			●	●	∧	∧	●
SUB		×	×	×		×	×	×			●	●	∧	∧	∧
SWI	×										●	1	●	●	●
TAX	×										●	●	●	●	●
TST	×		×			×	×				●	●	∧	∧	●
TXA	×										●	●	●	●	●
WAIT	×										●	●	●	●	●

Condition Code Symbols:

H	Half Carry (From Bit 3)	C	Carry/Borrow
I	Interrupt Mask	∧	Test and Set if True, Cleared Otherwise
N	Negative (Sign Bit)	●	Not Affected
Z	Zero	?	Load CC Register From Stack

Table 2-12 Operation Code Map

	Bit Manipulation		Branch	Read/Modify/Write					Control		Register/Memory						
	Test & Branch	Set/ Clear	Rel	DIR	A	X	,X1	,X0	IMP	IMP	IMM	DIR	EXT	,X2	,X1	,X0	
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	← HIGH
0	BRSET0	BSET0	BRA	NEG					RTI*	—	SUB						0
1	BRCLR0	BCLR0	BRN	—					RTS*	—	CMP						1
2	BRSET1	BSET1	BHI	—					—	—	SBC						2
3	BRCLR1	BCLR1	BLS	COM					SWI*	—	CPX						3
4	BRSET2	BSET2	BCC	LSR					—	—	AND						4
5	BRCLR2	BCLR2	BCS	—					—	—	BIT						5
6	BRSET3	BSET3	BNE	ROR					—	—	LDA						6
7	BRCLR3	BCLR3	BEQ	ASR					—	TAX*	—	STA				STA(+1)	7
8	BRSET4	BSET4	BHCC	LSL/ASL					—	CLC	EOR						8
9	BRCLR4	BCLR4	BHCS	ROL					—	SEC	ADC						9
A	BRSET5	BSET5	BPL	DEC					—	CLI*	ORA						A
B	BRCLR5	BCLR5	BMI	—					—	SEI*	ADD						B
C	BRSET6	BSET6	BMC	INC					—	RSP*	—	JMP(−1)					C
D	BRCLR6	BCLR6	BMS	TST(−1)	TST			TST(−1)	DAA*	NOP	BSR*	JSR(+2)		JSR(+1)		JSR(+2)	D
E	BRSET7	BSET7	BIL	—					STOP*	—	LDX						E
F	BRCLR7	BCLR7	BIH	CLR					WAIT*	TXA*	—	STX				STX(+1)	F
	3/5	2/5	2/3	2/5	1/2	1/2	2/6	1/5	1/*	1/1	2/2	2/3	3/4	3/5	2/4	1/3	

LOW

(NOTES) 1. "—" is an undefined operation code.

2. The lowermost numbers in each column represent a byte count and the number of cycles required (byte count/number of cycles). The number of cycles for the mnemonics asterisked (*) is as follows:

RTI	8	TAX	2
RTS	5	RSP	2
SWI	10	TXA	2
DAA	2	BSR	5
STOP	4	CLI	2
WAIT	4	SEI	2

3. The parenthesized numbers must be added to the cycle count of the particular instruction.

(8) Additional Instructions

The following new instructions are used on the HD6305X and HD6305Y family.

DAA Converts the contents of the accumulator into BCD code.

WAIT Causes the MCU to enter the wait mode. For this mode, see the topic, Wait Mode.

STOP Causes the MCU to enter the stop mode. For this mode, see the topic, Stop Mode.

2.13 Operation at Each Instruction Cycle

The HD6305X1/X2/Y1/Y2, HD63P05Y1 employs a mechanism of the pipeline control for the instruction fetch and the subsequent instruction fetch is performed during the current instruction being executed.

Table 2-13 provides the information about the relationship among each data on the Address Bus, Data Bus and R/$\overline{W}$ status in cycle-by-cycle basis during the execution of each instruction.

Table 2-13 Cycle-by-Cycle Operation

Address Mode & Instructions	Cycles	Cycle #	Address Bus	R/$\overline{W}$	Data Bus
IMMEDIATE					
ADC, ADD, AND, BIT, CMP, CPX, EOR, LDA, LDX, ORA, SBC, SUB	2	1	Op Code Address +1	1	Operand Data
		2	Op Code Address +2	1	Next Op Code
DIRECT					
ADC, ADD, AND, BIT, CMP, CPX, EOR, LDA, LDX, ORA, SBC, SUB	3	1	Op Code Address +1	1	Address of Operand
		2	Address of Operand	1	Operand Data
		3	Op Code Address +2	1	Next Op Code
STA, STX	3	1	Op Code Address +1	1	Address of Operand
		2	Address of Operand	0	Data from Acc. / Data from Ix.
		3	Op Code Address +1	1	Next Op Code
JMP	2	1	Op Code Address +1	1	Jump Address
		2	Jump Address	1	Next Op Code
JSR	5	1	Op Code Address +1	1	Jump Address (LSB)
		2	1FFF	1	Irrelevant Data
		3	Stack Pointer	0	Return Address (LSB)
		4	Stack Pointer -1	0	Return Address (MSB)
		5	Jump Address	1	First Subroutine Op Code
ASR, CLR, COM, DEC, INC, LSL, LSR, NEG, ROL, ROR	5	1	Op Code Address +1	1	Address of Operand
		2	Address of Operand	1	Operand Data
		3	1FFF	1	Irrelevant Data
		4	Address of Operand	0	New Operand Data
		5	Op Code Address +2	1	Next Op Code
TST	4	1	Op Code Address +1	1	Address of Operand
		2	Address of Operand	1	Operand Data
		3	1FFF	1	Irrelevant Data
		4	Op Code Address +2	1	Next Op Code

(to be continued)

Address Mode & Instructions	Cycles	Cycle #	Address Bus	R/$\overline{W}$	Data Bus
EXTENDED					
ADC, ADD, AND,	4	1	Op Code Address +1	1	Address of Operand (MSB)
BIT, CMP, CPX,		2	Op Code Address +2	1	Address of Operand (LSB)
EOR, LDA, LDX,		3	Address of Operand	1	Operand Data
ORA, SBC, SUB		4	Op Code Address +3	1	Next Op Code
STA, STX	4	1	Op Code Address +1	1	Address of Operand (MSB)
		2	Op Code Address +2	1	Address of Operand (LSB)
		3	Address of Operand	0	Data from Acc. / Data from Ix.
		4	Op Code Address +3	1	Next Op Code
JMP	3	1	Op Code Address +1	1	Jump Address (MSB)
		2	Op Code Address +2	1	Jump Address (LSB)
		3	Jump Address	1	Next Op Code
JSR	6	1	Op Code Address +1	1	Jump Address (MSB)
		2	Op Code Address +2	1	Jump Address (LSB)
		3	1FFF	1	Irrelevant Data
		4	Stack Pointer	0	Return Address (LSB)
		5	Stack Pointer -1	0	Return Address (MSB)
		6	Jump Address	1	First Subroutine Op Code
INDEXED (No offset)					
ADC, ADD, AND,	3	1	Op Code Address +1	1	Next Op Code
BIT, CMP, CPX,		2	Ix	1	Operand Data
EOR, LDA, LDX,		3	Op Code Address +1	1	Next Op Code
ORA, SBC, SUB					
STA, STX	4	1	Op Code Address +1	1	Next Op Code
		2	1FFF	1	Irrelevant Data
		3	Ix	0	Data from Acc. / Data from Ix.
		4	Op Code Address +1	1	Next Op Code
JMP	2	1	Op Code Address +1	1	Next Op Code
		2	Ix	1	First Op Code of Jump Routine

(to be continued)

Address Mode & Instructions	Cycles	Cycle #	Address Bus	R/W	Data Bus
JSR	5	1	Op Code Address +1	1	Next Op Code
		2	1FFF	1	Irrelevant Data
		3	Stack Pointer	0	Return Address (LSB)
		4	Stack Pointer -1	0	Return Address (MSB)
		5	Ix	1	First Subroutine Op Code
ASR, CLR, COM,	5	1	Op Code Address +1	1	Next Op Code
DEC, INC, LSL,		2	Ix	1	Operand Data
LSR, NEG, ROL,		3	1FFF	1	Irrelevant Data
ROR		4	Ix	0	New Operand Data
		5	Op Code Address +1	1	Next Op Code
TST	4	1	Op Code Address +1	1	Next Op Code
		2	Ix	1	Operand Data
		3	1FFF	1	Irrelevant Data
		4	Op Code Address +1	1	Next Op Code
INDEXED (8-bit offset)					
ADC, ADD, AND,	4	1	Op Code Address +1	1	Offset
BIT, CMP, CPX,		2	1FFF	1	Irrelevant Data
EOR, LDA, LDX,		3	Ix + Offset	1	Operand Data
ORA, SBC, SUB		4	Op Code Address +2	1	Next Op Code
STA, STX	4	1	Op Code Address +1	1	Offset
		2	1FFF	1	Irrelevant Data
		3	Ix + Offset	0	{ Data from Acc. / Data from Ix.
		4	Op Code Address +2	1	Next Op Code
JMP	3	1	Op Code Address +1	1	Offset
		2	1FFF	1	Irrelevant Data
		3	Ix + Offset	1	First Op Code of Jump Routine
JSR	5	1	Op Code Address +1	1	Offset
		2	1FFF	1	Irrelevant Data
		3	Stack Pointer	0	Return Address (LSB)
		4	Stack Pointer -1	0	Return Address (MSB)
		5	Ix + Offset	1	First Subroutine Op Code
ASR, CLR, COM,	6	1	Op Code Address +1	1	Offset
DEC, INC, LSL,		2	1FFF	1	Irrelevant Data
LSR, NEG, ROL,		3	Ix + Offset	1	Operand Data
ROR		4	1FFF	1	Irrelevant Data
		5	Ix + Offset	0	New Operand Data
		6	Op Code Address +1	1	Next Op Code
TST	5	1	Op Code Address +1	1	Offset
		2	1FFF	1	Irrelevant Data
		3	Ix + Offset	1	Operand Data
		4	1FFF	1	Irrelevant Data
		5	Op Code Address +2	1	Next Op Code
INDEXED (16-bit offset)					
ADC, ADD, AND,	5	1	Op Code Address +1	1	Offset (MSB)
BIT, CMP, CPX,		2	Op Code Address +2	1	Offset (LSB)
EOR, LDA, LDX,		3	1FFF	1	Irrelevant Data
ORA, SBC, SUB		4	Ix + Offset	1	Operand Data
		5	Op Code Address +1	1	Next Op Code
STA, STX	5	1	Op Code Address +1	1	Offset (MSB)
		2	Op Code Address +2	1	Offset (LSB)
		3	1FFF	1	Irrelevant Data
		4	Ix + Offset	0	{ Data from Acc. / Data from Ix.
		5	Op Code Address +3	1	Next Op Code
JMP	4	1	Op Code Address +1	1	Offset (MSB)
		2	Op Code Address +2	1	Offset (LSB)
		3	1FFF	1	Irrelevant Data
		4	Ix + Offset	1	First Op Code of Jump Routine
JSR	6	1	Op Code Address +1	1	Offset (MSB)
		2	Op Code Address +2	1	Offset (LSB)
		3	1FFF	1	Irrelevant Data
		4	Stack Pointer	0	Return Address (LSB)
		5	Stack Pointer -1	0	Return Address (MSB)
		6	Ix + Offset	1	First Subroutine Op Code

(to be continued)

Address Mode & Instructions	Cycles	Cycle #	Address Bus	R/$\overline{W}$	Data Bus
IMPLIED					
ASR, CLR, COM, DEC, INC, LSL, LSR, NEG, ROL, ROR, TST	2	1	Op Code Address +1	1	Next Op Code
		2	Op Code Address +1	1	Next Op Code
CLC, NOP, SEC	1	1	Op Code Address +1	1	Next Op Code
RSP, TAX, TXA	2	1	Op Code Address +1	1	Next Op Code
		2	Op Code Address +1	1	Next Op Code
CLI, SEI	2	1	Op Code Address +1	1	Next Op Code
		2	1FFF	1	Irrelevant Data
DAA	2	1	Op Code Address +1	1	Next Op Code
		2	Op Code Address +1	1	Next Op Code
STOP, WAIT	4	1	Op Code Address +1	1	Next Op Code
		2	1FFF	1	Irrelevant Data
		3	1FFF	1	Irrelevant Data
		4	Op Code Address +1	1	Next Op Code
RTI	8	1	Op Code Address +1	1	Next Op Code
		2	1FFF	1	Irrelevant Data
		3	Stack Pointer	1	CC
		4	Stack Pointer +1	1	Acc.
		5	Stack Pointer +2	1	Ix.
		6	Stack Pointer +3	1	Return Address (MSB)
		7	Stack Pointer +4	1	Return Address (LSB)
		8	Return Address	1	First Op Code of Return Routine
RTS	5	1	Op Code Address +1	1	Next Op Code
		2	1FFF	1	Irrelevant Data
		3	Stack Pointer	1	Return Address (MSB)
		4	Stack Pointer +1	1	Return Address (LSB)
		5	Return Address	1	First Op Code of Return Routine
SWI	10	1	Op Code Address +1	1	Next Op Code
		2	1FFF	1	Irrelevant Data
		3	Stack Pointer	0	Return Address (LSB)
		4	Stack Pointer-1	0	Return Address (MSB)
		5	Stack Pointer-2	0	Ix.
		6	Stack Pointer-3	0	Acc.
		7	Stack Pointer-4	0	CC
		8	Vector Address 1FFC	1	Address of SWI Routine (MSB)
		9	Vector Address 1FFD	1	Address of SWI Routine (LSB)
		10	Address of SWI Routine	1	First Op Code of SWI Routine
RELATIVE					
BCC, BCS, BEQ, BHCC, BHCS, BHI, BIH, BIL, BLS, BMC, BMI, BMS, BNE, BPL, BRA, BRN	3	1	Op Code Address +1	1	Next Op Code
		2	1FFF	1	Irrelevant Data
		3	(Branch Address .....Test = "1" Op Code Address +1 .... Test = "0"	1	(First Op Code of Branch Routine Next Op Code
BSR	5	1	Op Code Address +1	1	Offset
		2	1FFF	1	Irrelevant Data
		3	Stack Pointer	0	Return Address (LSB)
		4	Stack Pointer-1	0	Return Address (MSB)
		5	Branch Address	1	First Op Code of Subroutine
BIT TEST AND BRANCH					
BRCLR, BRSET	5	1	Op Code Address +1	1	Address of Operand
		2	Address of Operand	1	Operand Data
		3	Op Code Address +2	1	Offset
		4	1FFF	1	Irrelevant Data
		5	(Branch Address .....Test ="1" Op Code Address +3 .....Test = "0"	1	(First Op Code of Branch Address Next Op Code
BIT SET/CLEAR					
BCLR, BSET	5	1	Op Code Address +1	1	Address of Operand
		2	Address of Operand	1	Operand Data
		3	1FFF	1	Irrelevant Data
		4	Address of Operand	0	New Operand Data
		5	Op Code Address +1	1	Next Op Code

3. INSTRUCTION SYSTEM

3.1 Symbols and Abbreviations

The meanings of symbols and abbreviations are shown below.

(1) Operation

( ) = contents

← = movement direction

+ = addition

− = subtraction

∧ = AND

∨ = OR

⊕ = Exclusive OR

$\bar{x}$ = NOT

(2) Register symbols in CPU

ACCA = accumulator A

CCR = condition codes register

IX = index register, 8 bits

PC = program counter, 14 bits

PCH = the six most significant bits of program counter

PCL = the eight least significant bits of program counter

SP = stack pointer, 5 bits

(3) Memory and addressing codes

M = stored address

MH = the eight most significant bits of stored address

ML = the eight least significant bits of stored address

M+1 = stored address M plus 1

Msp = stored address indicated by stack pointer

Imm = immediate value

Disp = displacement value = M − (IX)

D = displacement value = M − (IX)

DH = displacement value = the eight most significant bits

DL = displacement value = the eight least significant bits

Rel = relative value

IMPLIED = implied addressing

RELATIVE = relative addressing

ACCUMULATOR = accumulator addressing

INDEX REG. = index register addressing

IMMEDIATE = immediate addressing

DIRECT = direct addressing

EXTENDED = extended addressing

INDEXED 0 BYTE OFFSET = indexed addressing 0 byte offset

INDEXED 1 BYTE OFFSET = indexed addressing 1 byte offset

INDEXED 2 BYTE OFFSET = indexed addressing 2 byte offset

EA = effective address

(4) Contents of bits 0 through 4 of condition codes register

C =	carry - borrow	bit 0
Z =	zero	bit 1
N =	negative	bit 2
I =	interrupt mask	bit 3
H =	half carry from bit 3 to bit 4	bit 4

(5) Status of each bit before execution of instruction

An = bit n of ACCA (n = 7, 6, 5, ..., 0)

Mn = bit n of M (n = 7, 6, 5, ..., 0)

Xn = bit n of IX (n = 7, 6, 5, ..., 0)

(6) Status of each bit on result after execution of instruction

Rn = bit n of result (n = 7, 6, 5, ..., 0)

(7) Symbols on instruction's format

P = each addressing mode on Immediate, Direct, Extended and index of 0, 1 and 2 byte offset

Q = each addressing mode on Direct and index of 0 and 1 byte offset

A = accumulator addressing mode

X = index register addressing mode

DR = direct addressing mode

dd = relative operand (8 bits)

n = bit n of memory (n = 7, 6, 5, ..., 0)

(8) Status of HD6305's interrupt pin

$\overline{\text{INT}}$ = status of interrupt pin (high, low)

Arithmetic Operation
ADC

ADC (ADd with Carry)

Format

ADC P

Operation

ACCA ← (ACCA) + (M) + (C)

Condition Codes

H: Set if a carry occurs from bit 3; otherwise reset.
I: Not affected.
N: Set if the most significant bit of the result is set; reset otherwise.
Z: Set if the result is 0; otherwise reset.
C: Set if a carry occurs from the most significant bit of the result; otherwise reset.

Description

Adds the contents of the carry bit C to the sum of the contents of ACCA and M and stores the result into ACCA.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
IMMEDIATE	ADC	#Imm	A9	Imm		2	2
DIRECT	ADC	M	B9	M		2	3
EXTENDED	ADC	M	C9	MH	ML	3	4
INDEXED 0 BYTE OFFSET	ADC	0,X	F9			1	3
INDEXED 1 BYTE OFFSET	ADC	Disp,X	E9	D		2	4
INDEXED 2 BYTE OFFSET	ADC	Disp,X	D9	DH	DL	3	5

Example

```
0100 B6 02      LDA     VAL2      (EXVAL5,EXVAL6)+(VAL1,VAL2)
0102 CB 0006    ADD     EXVAL6    *          =(EXVAL5, EXVAL6)
0105 C7 0006    STA     EXVAL6    *
0108 B6 01      LDA     VAL1      *
010A C9 0005    ADC     EXVAL5    ***
010D C7 0005    STA     EXVAL5    *
```

Arithmetic Operation
ADD

ADD (ADD without carry)

Format

ADD P

Operation

ACCA ← (ACCA) + (M)

Condition Codes

H: Set if a carry occurs from bit 3 otherwise reset.
I: Not affected.
N: Set if the significant bit of the result is 1; otherwise reset.
Z: Set if the result is 0; otherwise reset.
C: Set if a carry occurs from the most significant bit of the result; otherwise reset.

Description

Adds the M contents to ACCA contents, and stores the result into ACCA.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
IMMEDIATE	ADD	#Imm	AB	Imm		2	2
DIRECT	ADD	M	BB	M		2	3
EXTENDED	ADD	M	CB	MH	ML	3	4
INDEXED 0 BYTE OFFSET	ADD	0,X	FB			1	3
INDEXED 1 BYTE OFFSET	ADD	Disp,X	EB	D		2	4
INDEXED 2 BYTE OFFSET	ADD	Disp,X	DB	DH	DL	3	5

Example

```
0110 B6 10      LDA     VAL1    (VAL1)+(WORK)=(RESULT)
0112 BB FF      ADD     WORK    *
0114 B7 50      STA     RESULT  *
```

Logical Operation
AND

AND (logical AND)

Format

AND P

Condition Codes

H: Not affected.
I: Not affected.
N: Set if the most significant bit of the result is "1"; otherwise reset.
Z: Set if the result is "0"; otherwise reset.
C: Not affected.

Operation

ACCA ← (ACCA) ∧ (M)

Description

Performs logical AND between the ACCA contents and the M contents, and stores the result into ACCA.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
IMMEDIATE	AND	#Imm	A4	Imm		2	2
DIRECT	AND	M	B4	M		2	3
EXTENDED	AND	M	C4	MH	ML	3	4
INDEXED 0 BYTE OFFSET	AND	0,X	F4			1	3
INDEXED 1 BYTE OFFSET	AND	Disp,X	E4	D		2	4
INDEXED 2 BYTE OFFSET	AND	Disp,X	D4	DH	DL	3	5

Example

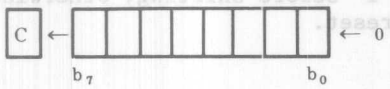
```
0107 F6          LDA     0,X     ERASE UPPER 4 BITS
0108 A4 0F       AND     #$0F    *
010A F7          STA     0,X     (RESTORE)
010B 5C          INC     X       *
010C 20 F2       BRA     LOOP    *
```

ASL (Arithmetic Shift Left)

Format

```
ASL Q
ASL A
ASL X
```

Operation



Condition Codes

H: Not affected.
I: Not affected.
N: Set if the most significant bit of the result is "1"; otherwise reset.
Z: Set if the result is "0"; otherwise reset.
C: Set if the most significant bit is "1" before shifting; otherwise reset.

Description

Shifts the contents of ACCA, IX or M 1 bit to the left. The bit 0 is loaded with "0". The carry bit C is loaded with the bit 7 of ACCA, IX or M.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
ACCUMULATOR	ASL	A	48			1	2
INDEX REG.	ASL	X	58			1	2
DIRECT	ASL	M	38	M		2	5
INDEXED 0 BYTE OFFSET	ASL	0,X	78			1	5
INDEXED 1 BYTE OFFSET	ASL	Disp,X	68	D		2	6

Example

```
010E B6 FF          LDA     WORK
0110 48     CHECK   ASL     A          BRANCH FOLLOWING BIT
0111 25 2E          BCS     BITON      *      7-6-5-4-3-2-1-0
     0113   BITOFF  EQU     *
0113 AE 64          LDX     #100
```

Shift and Rotation

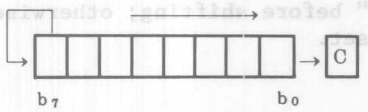
ASR

ASR (Arithmetic Shift Right)

Format

ASR Q
ASR A
ASR X

Operation



Condition Codes

H: Not affected.
I: Not affected.
N: Set if the most significant bit of the result is "1"; otherwise reset.
Z: Set if the result is "0"; otherwise reset.
C: Set if the most significant bit is "1" before shifting; otherwise reset.

Description

Shifts the contents of ACCA, IX or M 1-bit to the right. The bit 7 is not affected. The bit 0 is loaded into the carry bit C.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
ACCUMULATOR	ASR	A	47			1	2
INDEX REG.	ASR	X	57			1	2
DIRECT	ASR	M	37	M		2	5
INDEXED 0 BYTE OFFSET	ASR	0,X	77	D		1	5
INDEXED 1 BYTE OFFSET	ASR	Disp,X	67			2	6

Example

```
0143 37 FF      ASR   WORK      BRANCH OPTION (KEEPING BIT 7)
0145 25 17      BCS   OPT0
0147 37 FF      ASR   WORK
0149 25 1B      BCS   OPT1
014B 37 FF      ASR   WORK
014D 25 46      BCS   OPT2
```

BCC (Branch if Carry Clear)

Format

BCC dd

Condition Codes

Not affected.

Operation

PC ← (PC)+0002+Rel if (C)=0

Description

Tests the state of the C bit and causes a branch if C is "0".

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
RELATIVE	BCC	Rel	24	Rel		2	3

Example

```
014F B6 20          LDA     VAL2
0151 CB 0600        ADD     EXVAL6
0154 24 11          BCC     NORMAL    KETA  AGARI  NASHI
                    *
0156 5C             INC     X         KETA  AGARI
```

Bit Control

BCLR

BCLR (Bit CLeaR bit n)

Format

BCLR n, DR

Condition Codes

Not affected.

Operation

Mn ← 0

Description

Clears the bit n (n = 0 through 7) of M. The other bits are unaffected.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
DIRECT	BCLR	0,M	11	M		2	5
DIRECT	BCLR	1,M	13	M		2	5
DIRECT	BCLR	2,M	15	M		2	5
DIRECT	BCLR	3,M	17	M		2	5
DIRECT	BCLR	4,M	19	M		2	5
DIRECT	BCLR	5,M	1B	M		2	5
DIRECT	BCLR	6,M	1D	M		2	5
DIRECT	BCLR	7,M	1F	M		2	5

Example

```
0157 B6 03      LDA     CNTRL     ** MAKE CONTROL CODE **
0159 A4 F0      AND     #$F0      *
015B BA FF      ORA     WORK      *
015D B7 03      STA     CONTRL    *
015F 11 03      BCLR    0,CNTRL   CLEAR BIT 0,6,7 ABSOLUTELY
0161 1D 03      BCLR    6,CNTRL
0163 1F 03      BCLR    7,CNTRL
```

BCS (Branch if Carry Set)

Format

BCS dd

Condition Codes

Not affected.

Operation

PC ← (PC)+0002+Rel if (C)=1

Description

Tests the C-bit state and causes a branch if C is "1".

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
RELATIVE	BCS	Rel	25	Rel		2	3

Example

```
0165 B6 10          LDA    VAL1
0167 CB 0600        ADD    EXVAL6
016A 25 13          BCS    ABNML    KETA AGARI
                    *
016C C7 0600        STA    EXVAL6   KETA AGARI NASHI
```

Conditional Branch
BEQ

BEQ (Branch of EQual)

Format

BEQ dd

Condition Codes

Not affected.

Operation

PC←(PC)+0002+Rel if (Z)=1

Description

Tests the Z-bit state and causes a branch if Z is "1".

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
RELATIVE	BEQ	Rel	27	Rel		2	3

Example

```
016F B6 FF          LDA     WORK
0171 27 18          BEQ     AAAA      WORK = 0
0173 B1 50          CMP     RESULT
0175 27 28          BEQ     BBBB      WORK = RESULT
```

BHCC (Branch if Half Carry Clear)

Format

BHCC dd

Condition Codes

Not affected.

Operation

PC ← (PC)+0002+Rel if (H)=0

Description

Tests the H-bit state and causes a branch if H is "0".

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
RELATIVE	BHCC	Rel	28	Rel		2	3

Example

```
01E7 A1 09              CMP    #$9
01E9 23 02              BLS    DAALOW    $99 --->  INPUT
01EB AE 60     DAAH6    LDX    #$60      HIGH NYBLE NEEDS CORRECTION
               *
01ED 28 15     DAALOW   BHCC   DAAL9
01EF 9F                 TXA
```

Conditional Branch
BHCS

BHCS (Branch if Half Carry Set)

Format

BHCS dd

Condition Codes

Not affected.

Operation

PC ← (PC) + 0002 + Rel if (H) = 1

Description

Tests the H-bit state and causes a branch if H is "1".

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
RELATIVE	BHCS	Rel	29	Rel		2	3

Example

```
01F0 A1 09            CMP     #$9
01F2 23 02            BLS     DAALW1          $99 --- INPUT
01F4 AE 60      DAAH7 LDX     #$60            HIGH NYBLE NEEDS CORRECTION
                *
01F6 29 16      DAALW1 BHCS   DAAL6
01F8 A4 0F            AND     #$F
```

Conditional Branch

BHI

BHI (Branch if HIgher)

Format

BHI dd

Condition Codes

Not affected.

Operation

PC ← (PC)+0002+Rel if (C ∨ Z)=0
i.e. if (ACCA) > (M)
(unsigned binary numbers)

Description

Causes a branch if both C-bit and Z-bit are "0".
When the BHI instruction is executed immediately after either CMP or SUB instruction has been executed, a branch occurs if the minuend represented by the unsigned binary number (i.e. ACCA) is greater than the subtrahend represented by unsigned binary number (i.e. M).

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
RELATIVE	BHI	Rel	22	Rel		2	3

Example

```
0177 B6 10          LDA     VAL1
0179 B1 20          CMP     VAL2
017B 22 16          BHI     ZIP25       VAL1 > VAL2 (IGNORE SIGN BIT)
                    *
017D B7 FF          STA     WORK        VAL1 --> WORK (LOWER OR SAME)
```

Conditional Branch
BHS

BHS (Branch if Higher or Same)

Format

BHS dd

Condition Codes

Not affected.

Operation

PC ← (PC)+0002+Rel if (C)=0

Description

When the BHS instruction is executed after comparing or subtracting unsigned binary, if causes a branch if the register contents are greater than or equal to the M contents.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
RELATIVE	BHS	Rel	24	Rel		2	3

Example

```
0100 B6 10            LDA      VAL1
0102 B1 20            CMP      VAL2
0104 24 16            BHS      ZIP26     VAL1 >= VAL2  IGNORE SIGN BIT
                      *
0106 B7 FF            STA      WORK      VAL1 ---> WORK (LOWER)
```

Conditional Branch

BIH

BIH (Branch if Interrupt line is High)

Format

BIH dd

Condition Codes

Not affected.

Operation

PC ← (PC)+0002+Rel

if $\overline{INT}$=1 (high)

Description

Tests the external interrupt pin ($\overline{INT}$) state and causes a branch if it is high.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
RELATIVE	BIH	Rel	2F	Rel		2	3

Example

```
01C6 2F 04          BIH     INTH0   INT LINE CHECK
01C8 A6 28   INTL1  LDA     #$28    OUTPUT DATA = $28
01CA 20 02          BRA     NEXT2
01CC A6 FF   INTH0  LDA     #$FF    OUTPUT DATA = $FF
01CE C7 06E0 NEXT2  STA     PIA     OUTPUT
```

Conditional Branch

BIL

BIL (Branch if Interrupt line is Low)

Format

BIL dd

Condition Codes

Not affected.

Operation

PC ← (PC)+0002+Rel if $\overline{INT}$=0 (low)

Description

Tests the external interrupt pin ($\overline{INT}$) state and causes a branch if it is low.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
RELATIVE	BIL	Rel	2E	Rel		2	3

Example

```
01D1 2E 04          BIL     INTL2   INT LINE CHECK
01D3 A6 45    INTH3 LDA     #$45    OUTPUT DATA = $45
01D5 20 02          BRA     NEXT4
01D7 A6 00    INTL2 LDA     #$0     OUTPUT DATA = $00
01D9 C7 06E0  NEXT4 STA     PIA     OUTPUT
```

Logical Operation
BIT

BIT (BIt Test)

Format

BIT P

Operation

(ACCA)∧(M)

Condition Codes

H: Not affected.
I: Not affected.
N: Set if the most significant bit of the result of the AND operand is 1; otherwise cleared.
Z: Set if all the bits of the result of the AND operand are 0; otherwise cleared.
C: Not affected.

Description

Performs the logical AND operation between the ACCA contents and M contents and modifies the condition codes respectively. The ACCA contents and M contents are not affected.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
IMMEDIATE	BIT	#Imm	A5	Imm		2	2
DIRECT	BIT	M	B5	M		2	3
EXTENDED	BIT	M	C5	MH	ML	3	4
INDEXED 0 BYTE OFFSET	BIT	0,X	F5			1	3
INDEXED 1 BYTE OFFSET	BIT	Disp,X	E5	D		2	4
INDEXED 2 BYTE OFFSET	BIT	Disp,X	D5	DH	DL	3	5

Example

```
0400 B6 10      EVBIT   LDA     VAL1
0402 A5 F8              BIT     #$F8
0404 27 19              BEQ     OK          0 <= BIT ASSIGN (VAL1) <= 7
                *
0406 A6 E3      NG      LDA     #227        SET ERROR NUMBER
0408 CC 0432            JMP     ERROR
```

Conditional Branch
BLO

BLO (Branch if LOwer)

Format

BLO dd

Condition Codes

Not affected.

Operation

PC ← (PC)+0002+Rel if (C)=1

Description

Causes a branch when executing BLO after compare if register contents are less than M contents.
BLO is equivalent to BCS.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
RELATIVE	BLO	Rel	25	Rel		2	3

Example

```
040B B6 10              LDA     VAL1
040D B1 20              CMP     VAL2
040F 25 16              BLO     ZIP27     VAL1 < VAL2 (IGNORE SIGN BIT)
                *
0411 B7 FF              STA     WORK      VAL1 --> WORK  HIGHER OR SAME
```

Conditional Branch

BLS

BLS (Branch if Lower or Same)

Format

BLS dd

Condition Codes

Not affected.

Operation

PC ← (PC)+0002+Rel if (C V Z)=1

i.e. if (ACCA) $\leq$ (M)

Description

Causes a branch if either C-bit or Z-bit is "1". When the BHI instruction is executed immediately after either CMP or SUB instruction has been executed, a branch occurs if the minuend represented by the unsigned binary number (i.e. ACCA) is less than or equal to the subtrahend represented by unsigned binary number (i.e. M).

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
RELATIVE	BLS	Rel	23	Rel		2	3

Example

```
0413 B6 10      LDA     VAL1
0415 B1 20      CMP     VAL2
0417 23 16      BLS     ZIP28       VAL1 <= VAL2  IGNORE SIGN BIT
                *
0419 B7 FF              STA     WORK        VAL1 ---> WORK (HIGHER)
```

Conditional Branch

BMC

BMC (Branch if interrupt Mask is Clear)

Format

BMC dd

Condition Codes

Not affected.

Operation

PC ← (PC)+0002+Rel if (I)=0

Description

Tests the I-bit state and causes a branch if I is "0".

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
RELATIVE	BMC	Rel	2C	Rel		2	3

Example

```
0217 2C 07            BMC     MSKOFF    INTMSK OFF ?
0219 2E 05            BIL     MSKOFF    INT LINE LOW ?
021B C6 06E0          LDA     PIA       READ DATA
021E B7 FF            STA     WORK
0220 81      MSKOFF   RTS
```

Conditional Branch
BMI

BMI (Branch if MInus)

Format

BMI dd

Condition Codes

Not affected.

Operation

PC ← (PC)+0002+Rel if (N)=1

Description

Tests the N-bit state and causes a branch if N is "1".

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
RELATIVE	BMI	Rel	2B	Rel		2	3

Example

```
0425 B6 10          LDA     VAL1
0427 2B 16          BMI     ZIP29     VAL1 < 0
                    *
0429 B7 FF          STA     WORK      VAL1 ---> WORK (PLUS)
```

Conditional Branch
BMS

BMS (Branch if interrupt Mask is Set)

Format

BMS dd

Condition Codes

Not affected.

Operation

PC ← (PC)+0002+Rel if (I)=1

Description

Tests the I-bit state and causes a branch if I is "1".

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
RELATIVE	BMS	Rel	2D	Rel		2	3

Example

```
0221 2D 01          BMS     MSKON1    INTMSK ON ?
0223 81     MSKOF1  RTS               NO
0224 2D FD  MSKON1  BIL     MSKOF1    INT LINE LOW ?
0226 C6 06E0        LDA     PIA       DATA
0229 B7 FF          STA     WORK
022B 81             RTS
```

BNE (Branch if Not Equal)

Format

BNE dd

Condition Codes

Not affected.

Operation

PC ← (PC)+0002+Rel if (Z)=0

Description

Tests the Z-bit state and causes a branch if Z is "0".
Following a compare or subtract instruction, BNE will cause a branch if the arguments were different.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
RELATIVE	BNE	Rel	26	Rel		2	3

Example

```
020D B6 FF              LDA   WORK
020F 26 18              BNE   CCCC    WORK NOT = 0
0211 B1 50              CMP   RESULT
0213 26 1D              BNE   DDDD    WORK NOT = RESULT
```

Conditional Branch

BPL

BPL (Branch if PLus)

Format

BPL dd

Condition Codes

Not affected.

Operation

PC ← (PC)+0002+Rel if (N)=0

Description

Tests the N-bit state and causes a branch if N is "0".

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
RELATIVE	BPL	Rel	2A	Rel		2	3

Example

```
0215 B6 10              LDA   VAL1
0217 2A 16              BPL   ZIP31   VAL >= 0
                *
0219 B7 FF              STA   WORK    VAL1 ---> WORK (MINUS)
```

Unconditional Branch

BRA

BRA (BRanch Always)

Format

BRA dd

Condition Codes

Not affected.

Operation

PC ← (PC)+0002+Rel

Description

Causes an unconditional branch to the address gain from the operation shown above.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
RELATIVE	BRA	Rel	20	Rel		2	3

Example

```
0100 C6 0500          LDA     EXVAL5
0103 B7 50            STA     RESULT
0105 20 1E            BRA     END01     BRANCH TO END01 ALWAYS
                      *
      0107    CHECK8  EQU     *
```

Conditional Branch

BRCLR

BRCLR (BRanch if bit n is CLeaR)

Format

BRCLR n, DR, dd

Condition Codes

H: Not affected.
I: Not affected.
N: Not affected.
Z: Not affected.
C: Set if (Mn)=1; otherwise reset.

Operation

PC ← (PC)+0003+Rel if (Mn)=0

Description

Tests the bit n (n = 0 through 7) of M and causes a branch if the contents of Mn are "0".

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
RELATIVE	BRCLR	0,M,Rel	01	M	Rel	3	5
RELATIVE	BRCLR	1,M,Rel	03	M	Rel	3	5
RELATIVE	BRCLR	2,M,Rel	05	M	Rel	3	5
RELATIVE	BRCLR	3,M,Rel	07	M	Rel	3	5
RELATIVE	BRCLR	4,M,Rel	09	M	Rel	3	5
RELATIVE	BRCLR	5,M,Rel	0B	M	Rel	3	5
RELATIVE	BRCLR	6,M,Rel	0D	M	Rel	3	5
RELATIVE	BRCLR	7,M,Rel	0F	M	Rel	3	5

Example

```
0107 B6 03            LDA     CNTRL       ** SET CONTROL CODE **
0109 A4 0F            AND     #$0F
010B BA FF            ORA     WORK
010D B7 03            STA     CNTRL
                *             ** ACTION **
010F 09 03 17         BRCLR   4,CNTRL,ENGINE
0112 0F 03 28         BRCLR   7,CNTRL,GASCHK
```

Unconditional Branch
BRN

BRN (BRanch Never)

Format

BRN dd

Condition Codes

Not affected.

Operation

PC ← (PC)+0002

Description

It does not cause a branch. BRN, which requires 2-byte and 3 cycle long, is the inverse of BRA. This instruction is sometimes available for debugging program.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
RELATIVE	BRN	Rel	21	Rel		2	3

Example

```
0115 EF 04          STX   4,X
                    *
0117 21 FE          BRN   *       ** DELAY **
0119 21 FE          BRN   *
011B 21 FE          BRN   *
011D 21 FE          BRN   *
```

Conditional Branch

BRSET

BRSET (BRanch if bit n is SET)

Format

BRSET n, DR, dd

Condition Codes

H: Not affected.
I: Not affected.
N: Not affected.
Z: Not affected.
C: Set if (Mn)=1; otherwise reset.

Operation

PC ← (PC)+0003+Rel if (Mn)=1

Description

Tests the bit n (n = 0 through 7) of M, and causes a branch if the Mn contents are "1".

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
RELATIVE	BRSET	0,M,Rel	00	M	Rel	3	5
RELATIVE	BRSET	1,M,Rel	02	M	Rel	3	5
RELATIVE	BRSET	2,M,Rel	04	M	Rel	3	5
RELATIVE	BRSET	3,M,Rel	06	M	Rel	3	5
RELATIVE	BRSET	4,M,Rel	08	M	Rel	3	5
RELATIVE	BRSET	5,M,Rel	0A	M	Rel	3	5
RELATIVE	BRSET	6,M,Rel	0C	M	Rel	3	5
RELATIVE	BRSET	7,M,Rel	0E	M	Rel	3	5

Example

```
011F B6 03          LDA     CNTRL     ** SET CONTROL CODE **
0121 A4 8E          AND     #$8E
0123 BA FF          ORA     WORK
0125 B7 03          STA     CNTRL
                *                     ** ACTION **
0127 00 03 17 PROC1 BRSET   0,CNTRL,OIL
012A 0E 03 28 PROC2 BRSET   7,CNTRL,GAS
```

Bit Control

BSET

BSET (Bit SET bit n)

Format

BSET n,DR

Condition Codes

Not affected.

Operation

Mn ← 1

Description

Sets the bit n (n = 0 through 7) of M. All other bits are unaffected.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
DIRECT	BSET	0,M	10	M		2	5
DIRECT	BSET	1,M	12	M		2	5
DIRECT	BSET	2,M	14	M		2	5
DIRECT	BSET	3,M	16	M		2	5
DIRECT	BSET	4,M	18	M		2	5
DIRECT	BSET	5,M	1A	M		2	5
DIRECT	BSET	6,M	1C	M		2	5
DIRECT	BSET	7,M	1E	M		2	5

Example

```
0100 B6 50          LDA     RESULT
0102 2A 04          BPL     PLUS
                    *                   (MINUS)
0104 14 03          BSET    2,CNTRL
0106 16 FF          BSET    3,WORK
     0108   PLUS    EQU     *
0108 B6 20          LDA     VAL2
```

Subroutine Control

BSR

BSR (Branch to SubRoutine)

Format

BSR dd

Condition Codes

Not affected.

Operation

PC ← (PC)+0002
Msp ← (PCL), SP ← (SP)-0001
Msp ← (PCH), SP ← (SP)-0001
PC ← (PC)+Rel

Description

The program counter is increased by "2". The less significant bites (8-bits) of the program counter contents are pushed onto the stack. Then, stackpointer is decreased by "1". The more significant bits (6-bits) of the program counter contents are pushed onto the stack, then stackpointer is decreased by "1". Then a branch occurs to the address specified by the program counter.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
RELATIVE	BSR	Rel	AD	Rel		2	5

Example

```
010A A6 3B          LDA     #$3B      ACCA = INTERFACE (0011 1011)
010C AD 18          BSR     HAND
                *
010E A6 1E          LDA     #$1E      ACCA = INTERFACE (0001 1110)
0110 AD 28          BSR     FING
```

Bit Control
CLC

CLC (CLear Carry)

Format

CLC

Condition Codes

H: Not affected.
I: Not affected.
N: Not affected.
Z: Not affected.
C: Reset.

Operation

C ← 0

Description

Resets the carry bit C in the condition code register.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
IMPLIED	CLC		98			1	1

Example

```
0100 26 F8          BNE     CHK83
0102 B7 50          STA     RESULT
0104 98             CLC                 RETURN CODE SET 'OK'
0105 81             RTS
                    *
                    *
```

Bit Control
CLI

CLI (CLear Interrupt mask)

Format

CLI

Condition Codes

H: Not affected.
I: Reset.
N: Not affected.
Z: Not affected.
C: Not affected.

Operation

I ← 0

Description

Resets the interrupt mask bit in the processor condition code register. This enables the microprocessor to service interrupts that occurred through an interrupt request from peripheral equipment.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of cycles
			Byte 1	Byte 2	Byte 3		
IMPLIED	CLI		9A			1	2

Example

```
01FA 9B              SEI          INTERRUPT DISABLE
01FB 9C              RSP          RESET STACK POINTER
01FC CD 06F0         JSR    SYSINZ SYSTEM INITIALIZE
01FF 9A              CLI          INTERRUPT ENABLE
```

Arithmetic Operation
CLR

CLR (CLeaR)

Format

```
CLR Q
CLR A
CLR X
```

Condition Codes

H: Not affected.
I: Not affected.
N: Reset.
Z: Set.
C: Not affected.

Operation

```
IX   ← 0
  or
ACCA ← 0
  or
M    ← 0
```

Description

The contents of IX, ACCA or M are replaced with "0".

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
ACCUMULATOR	CLR	A	4F			1	2
INDEX REG.	CLR	X	5F			1	2
DIRECT	CLR	M	3F	M		2	5
INDEXED 0 BYTE OFFSET	CLR	0,X	7F			1	5
INDEXED 1 BYTE OFFSET	CLR	Disp,X	6F	D		2	6

Example

```
                *                          ** INITIALIZE **
0106 3F 07              CLR       PNTR
0108 3F 08              CLR       PNTR+1
010A 7F                 CLR       0,X
010B 4F                 CLR       A
```

Comparison and Test

CMP

CMP (CoMPare)

Format

CMP P

Operation

(ACCA)-(M)

Condition Codes

H: Not affected.
I: Not affected.
N: Set if the most significant bit of the result of the subtraction is "1"; otherwise reset.
Z: Set if the result of the subtraction is 0; otherwise reset.
C: Set if the absolute value of memory is greater than that of the accumulator; otherwise reset.

Description

Compares the ACCA contents with M contents, and affects the condition codes that can be referred to by conditional branch instructions. Both operands are unaffected.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
IMMEDIATE	CMP	#Imm	A1	Imm		2	2
DIRECT	CMP	M	B1	M		2	3
EXTENDED	CMP	M	C1	MH	ML	3	4
INDEXED 0 BYTE OFFSET	CMP	0,X	F1			1	3
INDEXED 1 BYTE OFFSET	CMP	Disp,X	E1	D		2	4
INDEXED 2 BYTE OFFSET	CMP	Disp,X	D1	DH	DL	3	5

Example

```
0110 E6 07          LDA     PNTR,X
                    *
0112 A1 41          CMP     #'A
0114 27 1A          BEQ     SECTA     ACCA = 'A'
0116 A1 42          CMP     #'B
0118 27 2A          BEQ     SECTB     ACCA = 'B'
011A 20 F0          BRA     INPUT
```

Logical Operation
COM

COM (COMplement)

Format

```
COM Q
COM A
COM X
```

Condition Codes

H: Not affected.
I: Not affected.
N: Set if the most significant bit of the result is "1"; otherwise cleared.
Z: Set if the result is "0"; otherwise reset.
C: Set.

Operation

$IX \leftarrow (\overline{IX}) = \$FF-(IX)$
or
$ACCA \leftarrow (\overline{ACCA}) = \$FF-(ACCA)$
or
$M \leftarrow (\overline{M}) = \$FF-(M)$

Description

Replaces the contents of ACCA, IX or M with its 1's complement.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
ACCUMULATOR	COM	A	43			1	2
INDEX REG.	COM	X	53			1	2
DIRECT	COM	M	33	M		2	5
INDEXED 0 BYTE OFFSET	COM	0,X	73			1	5
INDEXED 1 BYTE OFFSET	COM	Disp,X	63	D		2	6

Example

```
                *
     011C       SUBIN   EQU     *
011C 5C                 INC     X
011D E6 07              LDA     PNTR,X
011F 43                 COM     A       MODIFY DATA (REVERSE)
0120 81                 RTS
                *
```

Comparison and Test

CPX

CPX (ComPare indeX register)

Format

CPX P

Operation

(IX)-(M)

Condition Codes

H: Not affected.
I: Not affected.
N: Set if the most significant bit of the result is "1"; otherwise reset.
Z: Set if the result is "0"; otherwise reset.
C: Set if the absolute value of the contents of the memory is greater than that of the contents of IX; otherwise reset.

Description

Compares the IX contents with M contents. The condition code can be collated by means of the next conditional branch instruction. Both operands are unaffected.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
IMMEDIATE	CPX	#Imm	A3	Imm		2	2
DIRECT	CPX	M	B3	M		2	3
EXTENDED	CPX	M	C3	MH	ML	3	4
INDEXED 0 BYTE OFFSET	CPX	0,X	F3			1	3
INDEXED 1 BYTE OFFSET	CPX	Disp,X	E3	D		2	4
INDEXED 2 BYTE OFFSET	CPX	Disp,X	D3	DH	DL	3	5

Example

```
0121 A6 CC        LDA     #$CC        ACCA = INTERFACE TO CR OR LF
0123 BE 07        LDX     PNTR
0125 A3 0D        CPX     #$0D
0127 27 18        BEQ     CR          CARRIAGE RETURN
0129 A3 0A        CPX     #$0A
012B 27 28        BEQ     LF          LINE FEED
```

Arithmetic Operation
DAA

DAA (Decimal Adjust Accumulator)

Format

DAA

Operation

Convert binary Add result into Binary Coded Decimal (BCD).

Condition Codes

H: Not affected.
I: Not affected.
N: Set if the most significant bit of result is "1"; otherwise reset.
Z: Set if the result is "0", otherwise reset.
C: Set or reset with the rule under which DAA and its previous ABA, ADD and ADC is converted into BCD.

Description

Bit Condition of bit C before DAA execution	Lower 4 bits (bit 4 ∿ 7)	Early H bit (Half Carry)	Lower 4 bits (bit 0 ∿ 3)	Value added to ACCA by DAA execution (hexadecimal)	Bit condition of bit C before DAA execution
0	0-9	0	0-9	00	0
0	0-8	0	A-F	06	0
0	0-9	1	0-3	06	0
0	A-F	0	0-9	60	1
0	9-F	0	A-F	66	1
0	A-F	1	0-3	66	1
0	0-2	0	0-9	60	1
0	0-2	0	A-F	66	1
0	0-3	1	0-3	66	1

Add 00, 06, 60 66 (heradecimal) to ACCA according to the table shown left.

If the BCD Add result by ADD and ADC instruction is in ACCA, bit C or bit H, DAA executes above function.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
IMPLIED	DAA		8D			1	2

Example

Arithmetic Operation

DEC

DEC (DECrement)

Format

```
DEC Q
DEC A
DEC X
```

Condition Codes

H: Not affected.
I: Not affected.
N: Set if the most significant bit of the result is "1"; otherwise reset.
Z: Set if the result is "0"; otherwise reset.
C: Not affected.

Operation

IX ← (IX)-01
or
ACCA ← (ACCA)-01
or
M ← (M)-01

Description

Subtracts "1" from the contents of ACCA, IX or M.
N and Z bits are set and reset according to the result of this operation.
The C-bit is not affected.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
ACCUMULATOR	DEC	A	4A			1	2
INDEX REG.	DEC	X	5A			1	2
DIRECT	DEC	M	3A	M		2	5
INDEXED 0 BYTE OFFSET	DEC	0,X	7A			1	5
INDEXED 1 BYTE OFFSET	DEC	Disp,X	6A	D		2	6

Example

```
                    *
012D 4A        LOOP23 DEC          ** MOVE **
                      A
012E 2B 07             BMI          NEXT
0130 FE                LDX          0,X       *
0131 DF 0100           STX          $100,X    *
0134 5C                INC          X         *
0135 20 F6             BRA          LOOP23
                    *
     0137      NEXT   EQU          *
```

Logical Operation
EOR

EOR (Exclusive OR)

Format

EOR P

Operation

ACCA ← (ACCA) ⊕ (M)

Condition Codes

H: Not affected.
I: Not affected.
N: Set if the most significant bit of the result is "1"; otherwise reset.
Z: Set if the result is "0"; otherwise reset.
C: Not affected.

Description

Performs the logical EXCLUSIVE OR between the ACCA contents and M contents and stores the result into ACCA.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
IMMEDIATE	EOR	#Imm	A8	Imm		2	2
DIRECT	EOR	M	B8	M		2	3
EXTENDED	EOR	M	C8	MH	ML	3	4
INDEXED 0 BYTE OFFSET	EOR	0,X	F8			1	3
INDEXED 1 BYTE OFFSET	EOR	Disp,X	E8	D		2	4
INDEXED 2 BYTE OFFSET	EOR	Disp,X	D8	DH	DL	3	5

Example

```
                *                       ** ARRANGE CONTROL CODE **
0137 B6 03              LDA     CNTRL     XXXX XXXX
0139 A8 99              EOR     #$99      1001 1001
013B B7 03              STA     CNTRL
013D 20 14              BRA     ACT01
```

Arithmetic Operation
INC

INC (INCrement)

Format

```
INC Q
INC A
INC X
```

Condition Codes

H: Not affected.
I: Not affected.
N: Set if the most significant bit of the result is "1"; otherwise reset.
Z: Set if the result is "0"; otherwise reset.
C: Not affected.

Operation

```
IX ← (IX)+01
  or
ACCA ← (ACCA)+01
  or
M ← (M)+01
```

Description

Adds "1" to the contents of ACCA, IX or M.
N and Z bits are set or reset according to the result of this operation.
The C bit is not affected.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
ACCUMULATOR	INC	A	4C			1	2
INDEX REG.	INC	X	5C			1	2
DIRECT	INC	M	3C	M		2	5
INDEXED 0 BYTE OFFSET	INC	0,X	7C			1	5
INDEXED 1 BYTE OFFSET	INC	Disp,X	6C	D		2	6

Example

```
0100 4C          LOOP3   INC     A           *
0101 A1 64               CMP     #100
0103 22 1B               BHI     EXIT        CHECK COUNTER (100 TIMES)
0105 FE                  LDX     0,X
0106 DF 0300             STX     $300,X      MOVE
0109 5C                  INC     X           *
010A 20 F4               BRA     LOOP3
```

Conditional Branch

JMP

JMP (JuMP)

Format

JMP P

Condition Codes

Not affected.

Operation

PC ← EA

Description

A jump occurs to the instruction stored at the effective address. The effective address is obtained according to the rules for EXTended, DIRect or INDexed addressing.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
DIRECT	JMP	M	BC	M		2	2
EXTENDED	JMP	M	CC	MH	ML	3	3
INDEXED 0 BYTE OFFSET	JMP	0,X	FC			1	2
INDEXED 1 BYTE OFFSET	JMP	Disp,X	EC	D		2	3
INDEXED 2 BYTE OFFSET	JMP	Disp,X	DC	DH	DL	3	4

Example

```
010C B6 10          LDA     VAL1
010E C7 0500        STA     EXVAL5
0111 B6 20          LDA     VAL2
0113 C7 0600        STA     EXVAL6
0116 CC 0333        JMP     END90       GO TO END-ROUTINE
```

Subroutine Control
JSR

JSR (Jump to SubRoutine)

Format

JSR P

Condition Codes

Not affected.

Operation

Note)
PC ← (PC)+n
Msp ← (PCL), SP ← (SP)-0001
Msp ← (PCH), SP ← (SP)-0001
PC ← EA

Description

The program counter is increased by "n"[Note)] in the addressing mode, then pushed onto the 2-byte stack. And the stack point is updated. A jump occurs to the specified address.
The effective address is obtained according to the rules for EXTended, DIRect or INDexed addressing.

Note) n is equal to 1, 2 or 3, according to the number of bytes in the instruction code. Refer to the addressing code and the number of MPU cycles shown below.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
DIRECT	JSR	M	BD	M		2	5
EXTENDED	JSR	M	CD	MH	ML	3	6
INDEXED 0 BYTE OFFSET	JSR	0,X	FD			1	5
INDEXED 1 BYTE OFFSET	JSR	Disp,X	ED	D		2	5
INDEXED 2 BYTE OFFSET	JSR	Disp,X	DD	DH	DL	3	6

Example

```
                *                ** MAIN ROUTINE **
      0119      START   EQU      *
0119  CD 0407           JSR      INTRTN      INITIALIZE
011C  CD 04E5           JSR      KBRTN       INPUT FROM KEY-BOARD
011F  CD 03AD           JSR      ANARTN      ANALYSE
0122  CD 053C           JSR      PRCRTN      PROCESS
0125  CC 06CB           JMP      ENDRTN      END
```

LDA (LoaD Accumulator)

Format

LDA P

Operation

ACCA ← (M)

Condition Codes

H: Not affected.
I: Not affected.
N: Set if the most significant bit of the result is "1"; otherwise reset.
Z: Set if the result is "0"; otherwise reset.
C: Not affected.

Description

Loads the contents of memory into the accumulator.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
IMMEDIATE	LDA	#Imm	A6	Imm		2	2
DIRECT	LDA	M	B6	M		2	3
EXTENDED	LDA	M	C6	MH	ML	3	4
INDEXED 0 BYTE OFFSET	LDA	0,X	F6			1	3
INDEXED 1 BYTE OFFSET	LDA	Disp,X	E6	D		2	4
INDEXED 2 BYTE OFFSET	LDA	Disp,X	D6	DH	DL	3	5

Example

```
0128 B6 10      LDA     VAL1
012A B7 FF      STA     WORK
012C F6         LDA     0,X
012D B7 50      STA     RESULT
012F A6 FF      LDA     #$FF
```

Load & Store

LDX

LDX (LoaD indeX register)

Format

LDX P

Operation

IX ← (M)

Condition Codes

H: Not affected.
I: Not affected.
N: Set if the most significant bit of IX is "1"; otherwise reset.
Z: Set if all the bits of IX of the result are "0"; otherwise reset.
C: Not affected.

Description

Loads the M contents into IX. The condition code is set according to data.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
IMMEDIATE	LDX	#Imm	AE	Imm		2	2
DIRECT	LDX	M	BE	M		2	3
EXTENDED	LDX	M	CE	MH	ML	3	4
INDEXED 0 BYTE OFFSET	LDX	0,X	FE			1	3
INDEXED 1 BYTE OFFSET	LDX	Disp,X	EE	D		2	4
INDEXED 2 BYTE OFFSET	LDX	Disp,X	DE	DH	DL	3	5

Example

```
0131 BE 10      LDX     VAL1
0133 BF FF      STX     WORK
0135 FE         LDX     0,X
0136 BF 50      STX     RESULT
0138 AE FF      LDX     #$FF
```

LSL (Logical Shift Left)

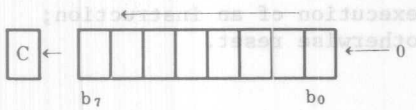
Format

LSL Q
LSL A
LSL X

Condition Codes

H: Not affected.
I: Not affected.
N: Set if the most significant bit of the result is "1"; otherwise reset.
Z: Set if the result is "0"; otherwise reset.
C: Set if the least significant bit of ACCA, IX or memory is "1" before execution of an instruction; otherwise reset.

Operation



Description

Shifts the contents of ACCA, IX or M 1-bit to the left. The bit 0 is loaded with "0". The carry bit C is loaded with the most significant bit of ACCA, IX or M.

Addressing Mode and Number of Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
ACCUMULATOR	LSL	A	48			1	2
INDEX REG.	LSL	X	58			1	2
DIRECT	LSL	M	38	M		2	5
INDEXED 0 BYTE OFFSET	LSL	0,X	78			1	5
INDEXED 1 BYTE OFFSET	LSL	Disp,X	68	D		2	6

Example

```
013A 38 FF        LSL     WORK    ** MULTIPLY  X 8 **
013C 38 FF        LSL     WORK
013E 38 FF        LSL     WORK
```

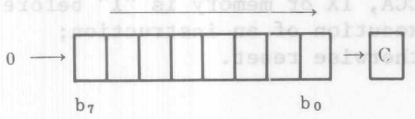
Shift & Rotation
LSR

LSR (Logical Shift Right)

Format

```
LSR Q
LSR A
LSR X
```

Operation



Condition Codes

H: Not affected.
I: Not affected.
N: Reset.
Z: Set if the result is "0"; otherwise reset.
C: Set if the least significant bit of ACCA, IX, or memory is "1" before execution of an instruction; otherwise reset.

Description

Shifts the contents of ACCA, IX or M 1-bit to the right. The bit 7 is loaded with "0". The carry bit C is loaded with the least significant bit of ACCA, IX or M.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
ACCUMULATOR	LSR	A	44			1	2
INDEX REG.	LSR	X	54			1	2
DIRECT	LSR	M	34	M		2	5
INDEXED 0 BYTE OFFSET	LSR	0,X	74			1	5
INDEXED 1 BYTE OFFSET	LSR	Disp,X	64	D		2	6

Example

```
0140 34 FF          LSR     WORK    ** DIVIDE / 16 **
0142 34 FF          LSR     WORK
0144 34 FF          LSR     WORK
0146 34 FF          LSR     WORK
```

Arithmetic Operation

NEG

NEG (NEGate)

Format

```
NEG Q
NEG A
NEG X
```

Condition Codes

H: Not affected.
I: Not affected.
N: Set if the most significant bit of the result is "1"; otherwise reset.
Z: Set if the result is "0"; otherwise reset.
C: Set if a borrow occurs; otherwise reset. Set if the contents of ACCA, IX or Memory are other than "0".

Operation

```
IX ← (IX)=00-(IX)
  or
ACCA ← (ACCA)=00-(ACCA)
  or
M ← (M)=00-(M)
```

Description

Replaces the contents of ACCA, IX or M with its two's complement, and stores ACCA or IX contents into M contents. Note that $80 (-128) is unaffected.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
ACCUMULATOR	NEG	A	40			1	2
INDEX REG.	NEG	X	50			1	2
DIRECT	NEG	M	30	M		2	5
INDEXED 0 BYTE OFFSET	NEG	0,X	70			1	5
INDEXED 1 BYTE OFFSET	NEG	Disp,X	60	D		2	6

Example

```
                *              CHECK RANGE (RELATIVE ADDRESSING)
0148 A1 81              CMP    #129         CHECK RANGE
014A 24 44              BCC    BERROR       *   BRANCH ERROR
014C 40                 NEG    A            OFFSET
014D 20 14              BRA    SET
                *
```

Unconditional Branch

NOP

NOP (No OPeration)

Format

NOP

Condition Codes

Not affected.

Operation

Description

This is a single-byte instruction which only causes the program counter to be increased. Other registers are unaffected.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
IMPLIED	NOP		9D			1	1

Example

```
014F 9D      NOP          ** DELAY **
0150 9D      NOP
0151 9D      NOP
0152 9D      NOP
0153 9D      NOP
0154 9D      NOP
```

Logical Operation

ORA

ORA (inclusive OR)

Format

ORA

Operation

ACCA ← (ACCA)v(M)

Condition Codes

H: Not affected.
I: Not affected.
N: Set if the most significant bit of the result is "1"; otherwise reset.
Z: Set if all the bits of the result are "0"; otherwise reset.
C: Not affected.

Description

Performs logical OR between ACCA contents and M contents, and stores the result into ACCA.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
IMMEDIATE	ORA	#Imm	AA	Imm		2	2
DIRECT	ORA	M	BA	M		2	3
EXTENDED	ORA	M	CA	MH	ML	3	4
INDEXED 0 BYTE OFFSET	ORA	0,X	FA			1	3
INDEXED 1 BYTE OFFSET	ORA	Disp,X	EA	D		2	4
INDEXED 2 BYTE OFFSET	ORA	Disp,X	DA	DH	DL	3	5

Example

```
0155 25 06              BCS     SKIP
                        *
     0157       ADCN    EQU     *       ** ADDITION CONTROL BIT **
0157 A6 14              LDA     #$14    0001 0100
0159 BA 03              ORA     CNTRL
015B B7 03              STA     CNTRL
     015D       SKIP    EQU     *
```

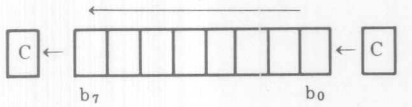
Shift & Rotation
ROL

ROL (ROtate Left)

Format

ROL Q
ROL A
ROL X

Operation



Condition Codes

H: Not affected.
I: Not affected.
N: Set if the most significant bit of the result is "1"; otherwise reset.
Z: Set of all the bits of the result are "0"; otherwise reset.
C: Set if the ACCA, IX, or the most significant bit of the memory is "1", before execution of an instruction; otherwise reset.

Description

Shifts the contents of ACCA, IX or M 1-bit to the left. The bit 0 is loaded with the carry bit C, while the carry bit C is loaded with the most significant bit of ACCA, IX or M.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
ACCUMULATOR	ROL	A	49			1	2
INDEX REG.	ROL	X	59			1	2
DIRECT	ROL	M	39	M		2	5
INDEXED 0 BYTE OFFSET	ROL	0,X	79			1	5
INDEXED 1 BYTE OFFSET	ROL	Disp,X	69	D		2	6

Example

```
                        *           ** REPEAT ACTION FOLLOWING CNTRL **
015D 98                 CLC
     015E       REPEAT  EQU  *
015E 39 03              ROL  CNTRL
0160 25 05              BCS  ACTION  ACTION & REPEAT OR ESCAPE
0162 9D                 NOP          *
0163 9D                 NOP          ** DELAY **
0164 9D                 NOP          *
0165 20 F7              BRA  REPEAT
```

ROR (ROtate Right)

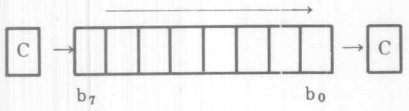
Format

ROR Q
ROR A
ROR X

Condition Codes

H: Not affected.
I: Not affected.
N: Set if the most significant bit of the result is "1"; otherwise reset.
Z: Set if all the bits of the result are "0"; otherwise reset.
C: Set if the ACCA, IX, or the least significant bit of Memory is "1"; otherwise reset.

Operation



Description

Shifts the contents of ACCA, IX or M one bit to the right. The bit 7 is loaded with the carry bit C, while the bit 0 is loaded with the carry bit C.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
ACCUMULATOR	ROR	A	46			1	2
INDEX REG.	ROR	X	56			1	2
DIRECT	ROR	M	36	M		2	5
INDEXED 0 BYTE OFFSET	ROR	0,X	76			1	5
INDEXED 1 BYTE OFFSET	ROR	Disp,X	66	D		2	6

Example

```
                *              ** REPEAT ACTION FOLLOWING CNTRL **
0167  98              CLC
      0168  REPT1     EQU     *
0168  36 03           ROR     CNTRL
016A  25 05           BCS     ACTN1   ACTION & REPEAT OR ESCAPE
016C  9D              NOP             *
016D  9D              NOP             ** DELAY  **
016E  9D              NOP             *
016F  20 F7           BRA     REPT1
```

Stack Pointer Operation
RSP

RSP (Reset Stack Pointer)

Format

RSP

Condition Codes

Not affected.

Operation

SP ← $FF

Description

Resets the stack pointer to the top ($FF) of the stack.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
IMPLIED	RSP		9C			1	2

Example

```
0200 9B          SEI              INTERRUPT DISABLE
0201 9C          RSP              RESET STACK POINTER
0202 CD 06F0     JSR     SYSINZ   SYSTEM INITIALIZE
0205 9A          CLI              INTERRUPT ENABLE
```

Interrupt Control

RTI

RTI (ReTurn from Interrupt)

Format

RTI

Condition Codes

Recovers the state saved onto the stack.

Operation

SP ← (SP)+0001, CCR ← (SP)
SP ← (SP)+0001, ACCA ← (SP)
SP ← (SP)+0001, IX ← (SP)
SP ← (SP)+0001, PCH ← (SP)
SP ← (SP)+0001, PCL ← (SP)

Description

Sets the stack contents indicated by SP to CCR, ACCA, IX, PCH, or PCL increasing SP by "1". Note that I = 0 when the interrupt mask bit of CCR saved onto the stack is "0".

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
IMPLIED	RTI		80			1	8

Example

```
020C CD 0345      JSR     KEYSCN     KEY INPUT
020F B7 10        STA     INKEY      STORE KEY CODE
0211 C0 0400      JSR     EXSWIN     INPUT EXTERNAL SW
0214 B7 11        STA     INSW       STORE SW CONDITION
0216 80           RTI                RETURN TO INTERRUPT
```

Subroutine Control

RTS

RTS (ReTurn from Subroutine)

Format

RTS

Condition Codes

Not affected.

Operation

SP ← (SP)+0001, PCH ← (SP)
SP ← (SP)+0001, PCL ← (SP)

Description

Increases SP by "1" and sets the address contents indicated by SP to more significant bits (6-bits) of PC. Increases SP with "1" again, and sets the address contents specified by SP to less significant bits (8-bits) of PC.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
IMPLIED	RTS		81			1	5

Example

```
0171 B7 FF      STA     WORK
0173 C6 0500    LDA     EXVALS
0176 B7 50      STA     RESULT
0178 98         CLC             RETURN CODE SET : OK
0179 81         RTS             *
                *
```

Arithmetic Operation
SBC

SBC (SuBtract with Carry)

Format

SBC P

Operation

ACCA ← (ACCA)-(M)-(C)

Condition Codes

H: Not affected.
I: Not affected.
N: Set if the most significant bit of the result is "1"; otherwise reset.
Z: Set if the result is "0"; otherwise reset.
C: Set if the absolute value of the contents of memory plus the carry bit C is greater than the absolute value of the contents of ACCA; otherwise reset.

Description

Subtracts the contents of M and the carry bit C from that of ACCA, and stores the result into ACCA.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
IMMEDIATE	SBC	#Imm	A2	Imm		2	2
DIRECT	SBC	M	B2	M		2	3
EXTENDED	SBC	M	C2	MH	ML	3	4
INDEXED 0 BYTE OFFSET	SBC	0,X	F2			1	3
INDEXED 1 BYTE OFFSET	SBC	Disp,X	E2	D		2	4
INDEXED 2 BYTE OFFSET	SBC	Disp,X	D2	DH	DL	3	5

Example

```
*                    (VAL1, VAL1+1)-(EXVAL5, EXVAL5+1)
*                                  =(EXVAL5, EXVAL5+1)
*
017A B6 11          LDA     VAL1+1   *
017C C0 0501        SUB     EXVAL5+1 *
017F C7 0501        STA     EXVAL5+1 *
0182 B6 10          LDA     VAL1     *
0184 C2 0500        SBC     EXVAL5   *
0187 C7 0500        STA     EXVAL5   *
```

Bit Control
SEC

SEC (SEt Carry)

Format

SEC

Operation

C bit ← 1

Condition Codes

H: Not affected.
I: Not affected.
N: Not affected.
Z: Not affected.
C: Set.

Description

Sets the carry bit C in the condition code register.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
IMPLIED	SEC		99			1	1

Example

```
018A 27 F8          BEQ   CHK84
018C B7 50          STA   RESULT
018E 99             SEC             RETURN CODE SET : NG
018F 81             RTS
                *
                *
```

Bit Control

SEI

SEI (SEt Interrupt mask)

Format

SEI

Condition Codes

H: Not affected.
I: Set.
N: Not affected.
Z: Not affected.
C: Not affected.

Operation

I bit ← 1

Description

Sets the interrupt mask bit I in the condition code register. If this bit is set, interrupt from peripheral equipment is disabled until the interrupt mask bit is cleared.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
IMPLIED	SEI		9B			1	2

Example

```
0206 9B          SEI          INTERRUPT DISABLE
0207 9C          RSP          RESET STACK POINTER
0208 CD 06F0     JSR   SYSINZ SYSTEM INITIALIZE
020B 9A          CLI          INTERRUPT ENABLE
```

Load & Store
STA

STA (STore Accumulator)

Format

STA P

Condition Codes

H: Not affected.
I: Not affected.
N: Set if the most significant bit of ACCA is "1"; otherwise reset.
Z: Set if the contents of ACCA are "0"; otherwise reset.
C: Not affected.

Operation

M ← (ACCA)

Description

Stores the ACCA contents into M. The ACCA contents are unaffected.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
DIRECT	STA	M	B7	M		2	3
EXTENDED	STA	M	C7	MH	ML	3	4
INDEXED 0 BYTE OFFSET	STA	0,X	F7			1	4
INDEXED 1 BYTE OFFSET	STA	Disp,X	E7	D		2	4
INDEXED 2 BYTE OFFSET	STA	Disp,X	D7	DH	DL	3	5

Example

```
0190 B6 10        LDA     VAL1
0192 B7 FF        STA     WORK
0194 B6 50        LDA     RESULT
0196 F7           STA     0,X
0197 A6 FF        LDA     #$FF
0199 D7 0500      STA     EXVAL5,X
```

Power Control
STOP

STOP (STOP)

Format

STOP

Condition Codes

Not affected.

Operation

Description

Enters into the STOP mode by STOP instruction
(For details, refer to STOP MODE in "2.9. Low Power Consumption Mode".)

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
IMPLIED	STOP		8E			1	4

Example

Load & Store
STX

STX (STore indeX register)

Format

STX P

Condition Codes

H: Not affected.
I: Not affected.
N: Set if the most significant bit of IX is "1"; otherwise reset.
Z: Set if the contents of IX are "0"; otherwise reset.
C: Not affected.

Operation

M ← (IX)

Description

Stores the IX contents into memory. IX contents are unaffected.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
DIRECT	STX	M	BF	M		2	3
EXTENDED	STX	M	CF	MH	ML	3	4
INDEXED 0 BYTE OFFSET	STX	0,X	FF			1	4
INDEXED 1 BYTE OFFSET	STX	Disp,X	EF	D		2	4
INDEXED 2 BYTE OFFSET	STX	Disp,X	DF	DH	DL	3	5

Example

```
019C BE 10      LDX     VAL1
019E BF FF      STX     WORK
01A0 BE 50      LDX     RESULT
01A2 FF         STX     0,X
01A3 AE FF      LDX     #$FF
01A5 DF 0500    STX     EXVAL5,X
```

Arithmetic Operation

SUB

SUB (SUBtract)

Format

SUB P

Operation

ACCA ← (ACCA)-(M)

Condition Codes

H: Not affected:
I: Not affected.
N: Set if the most significant bit of the result is "1"; otherwise reset.
Z: Set if the contents of the result are "0"; otherwise reset.
C: Set if the absolute value of the contents of memory is greater than the absolute value of the contents of ACCA; otherwise reset.

Description

Subtracts M contents from ACCA contents, and stores the result into ACCA.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
IMMEDIATE	SUB	#Imm	A0	Imm		2	2
DIRECT	SUB	M	B0	M		2	3
EXTENDED	SUB	M	C0	MH	ML	3	4
INDEXED 0 BYTE OFFSET	SUB	0,X	F0			1	3
INDEXED 1 BYTE OFFSET	SUB	Disp,X	E0	D		2	4
INDEXED 2 BYTE OFFSET	SUB	Disp,X	D0	DH	DL	3	5

Example

```
01A8 B6 10        LDA     VAL1    (VAL1)-(WORK)=(RESULT)
01AA B0 FF        SUB     WORK
01AC B7 50        STA     RESULT
```

Interrupt Control

SWI

SWI (SoftWare Interrupt)

Format

SWI

Condition Codes

H: Not affected.
I: Set.
N: Not affected.
Z: Not affected.
C: Not affected.

Operation

```
PC ← (PC)+0001
Msp ← (PCL),    SP ← (SP)-0001
Msp ← (PCH),    SP ← (SP)-0001
Msp ← (IX),     SP ← (SP)-0001
Msp ← (ACCA),   SP ← (SP)-0001
Msp ← (CCR),    SP ← (SP)-0001
I bit ← 1
PC ← (SWI interrupt vector address)
```

Description

All the registers other than the stack pointer (SP) are pushed onto the stack. The interrupt mask bit is then set. Performs vectoring to the address indicated by the contents of the SWI interrupt vector address.

Addressing Mode and Number of Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
IMPLIED	SWI		83			1	10

Example

```
01DC A6 FF          LDA     #$FF        *
01DE B7 36          STA     TIMER+1     * TIMER COUNTER SET
01E0 A6 3F          LDA     #$3F        *
01E2 B7 35          STA     TIMER       *
01E4 A6 03          LDA     #3          TIMER CODE SET
01E6 83             SWI                 MONITOR SERVICE CALL
```

Transfer
TAX

TAX (Transfer Accumulator to indeX register)

Format

TAX

Condition Codes

Not affected.

Operation

IX ← (ACCA)

Description

Transfers the ACCA contents to IX. The ACCA contents are unaffected.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
IMPLIED	TAX		97			1	2

Example

```
01AE 97          TAX       SAVE ACCUMULATOR
01AF A6 04       LDA  #4   *
01B1 BB 50       ADD  RESULT  ** ADD (RESULT+4)
01B3 B7 50       STA  RESULT  *
01B5 9F          TXA       REVIVE ACCUMULATOR
```

Comparison & Test
TST

TST (TeST)

Format

```
TST Q
TST A
TST X
```

Operation

```
(IX) - 00
    or
(ACCA) - 00
    or
(M) - 00
```

Condition Codes

H: Not affected.
I: Not affected.
N: Set if the most significant bit of ACCA, IX or M is "1"; otherwise reset.
Z: Set if the contents of ACCA, IX or M are "0"; otherwise reset.
C: Not affected.

Description

Sets N and Z bits of the condition code register according to the contents of ACCA, IX or M.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
ACCUMULATOR	TST	A	4D			1	2
INDEX REG.	TST	X	5D			1	2
DIRECT	TST	M	3D	M		2	4
INDEXED 0 BYTE OFFSET	TST	0,X	7D			1	4
INDEXED 1 BYTE OFFSET	TST	Disp,X	6D	D		2	5

Example

```
01BE 3D 03          TST     CNTRL
01C0 27 18          BEQ     INIT00   CNTRL=$00
                    *
01C2 3D FF          TST     WORK
01C4 2B 28          BMI     MINS00   WORK=(1XXX XXXX)
                    *
```

Transfer

TXA

TXA (Transfer indeX register to Accumulator)

Format

TXA

Condition Codes

Not affected.

Operation

ACCA ← (IX)

Description

Transfers the IX contents to ACCA. IX contents are unaffected.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
IMPLIED	TXA		9F			1	2

Example

```
01B6 97        TAX          SAVE ACCUMULATOR
01B7 A6 04     LDA   #4     *
01B9 BB 50     ADD   RESULT ** ADD (RESULT+4)
01BB B7 50     STA   RESULT *
01BD 9F        TXA          REVIVE ACCUMULATOR
```

Power Control
WAIT

WAIT (WAIT)

Format

WAIT

Condition Codes

Not affected.

Operation

Description

Enters into WAIT mode by WAIT instruction
(Refer to WAIT MODE in "2.9 Low Power Consumption Mode" for details).

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
IMPLIED	WAIT		8F			1	4

Example

4. APPLICATIONS AND PRECAUTIONS

4.1 Memory Space Expansion of HD6305X1/Y1, HD6305X2/Y2, HD63P05Y1

(1) ROM

ROM space can be expanded by externally connecting EPROM or Mask ROM to the MCU. Fig. 4-1 shows an example of using the HN482732A.

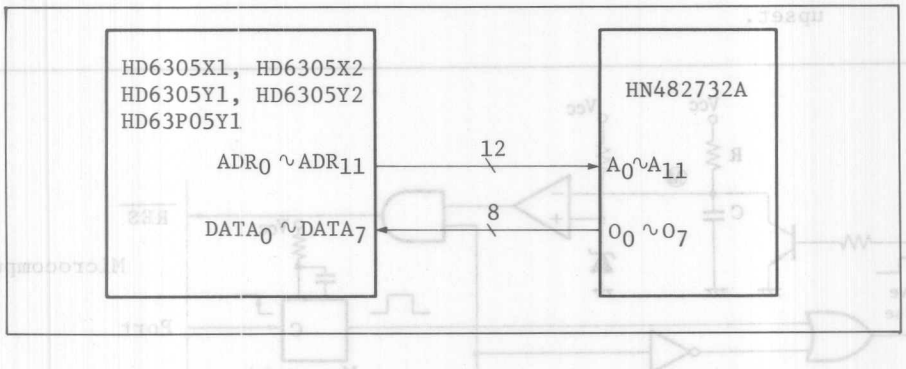


Fig. 4-1 Memory Space Expansion for ROM

(2) RAM

Fig. 4-2 shows a system configuration by using the HM6116.

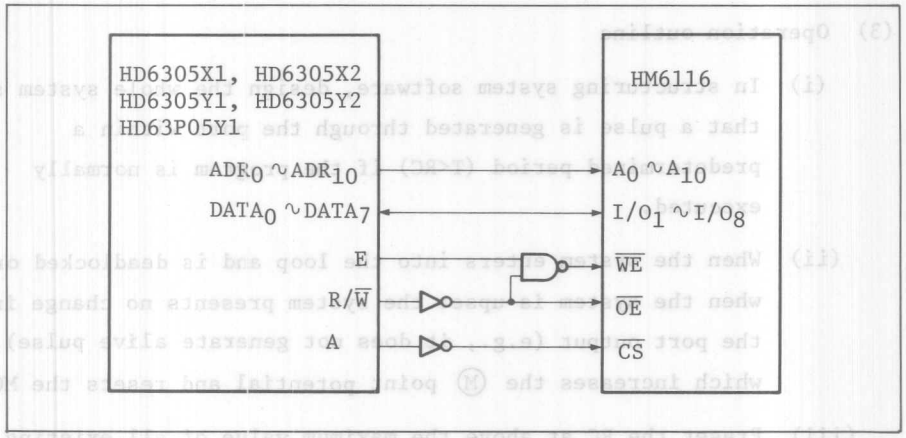


Fig. 4-2 Memory Space Expansion for RAM

4.2 Watch-Dog Timer

(1) Purpose

A simple method of implementing the watch dog timer, which is generally known as a means of recovery from a system upset, will be described below.

(2) Basic circuit

Fig. 4-3 shows the basic circuit for recovery from a system upset.

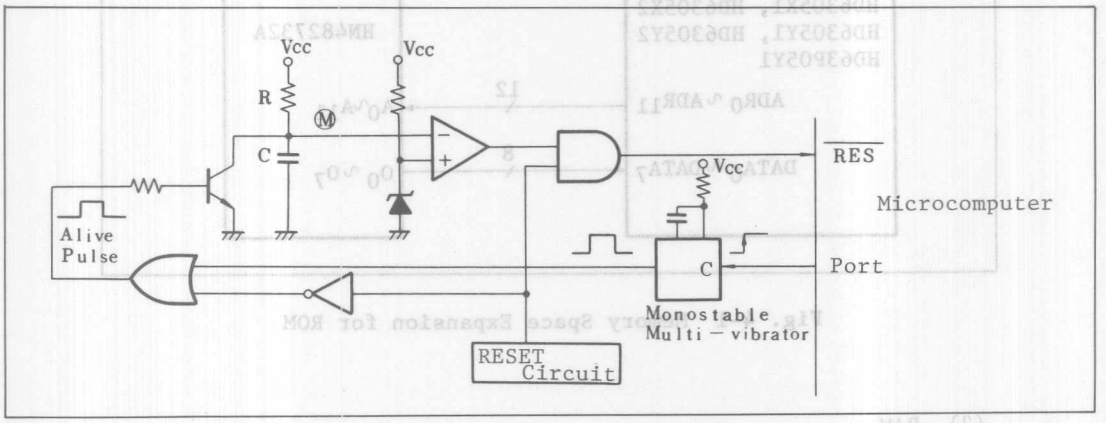


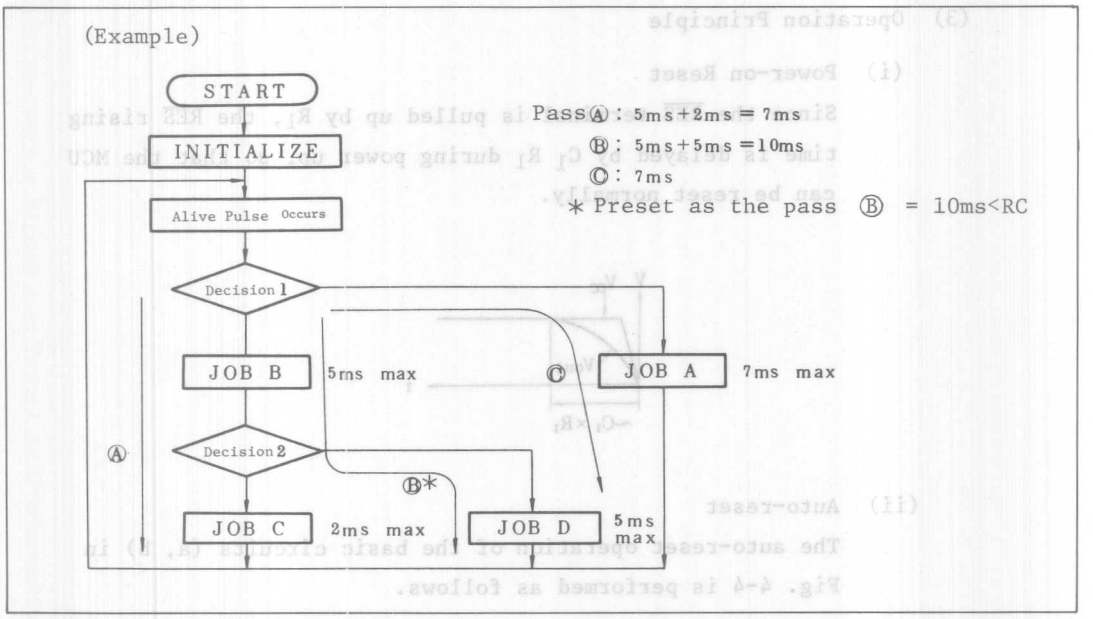
Fig. 4-3 Basic Circuit for Recovery from a System Upset

(3) Operation outline

(i) In structuring system software, design the whole system so that a pulse is generated through the port within a predetermined period (T<RC) if the program is normally executed.

(ii) When the system enters into the loop and is deadlocked or when the system is upset the system presents no change in the port output (e.g., it does not generate alive pulse), which increases the Ⓜ point potential and resets the MCU.

(iii) Preset the RC at above the maximum value of all existing routes in normal operation. RC is set to a value greater than 10ms in the example below.



4.3 Auto Reset Circuit

(1) Purpose

The following describes how to implement the reset circuit for power-on reset or auto reset at $V_{CC}<V_{LM}$. (V_{LM}; Treshold level.)

(2) Basic Circuit

Fig. 4-4 shows the basic circuit of auto reset.

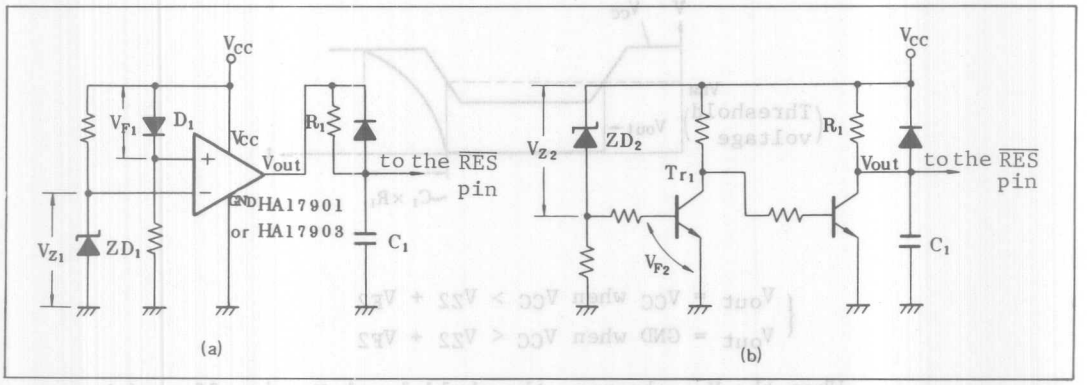
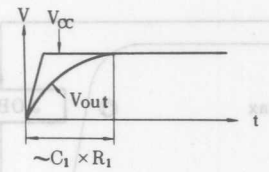


Fig. 4-4 Basic Circuit of Auto Reset

(3) Operation Principle

(i) Power-on Reset

Since the $\overline{RES}$ terminal is pulled up by R_1, the $\overline{RES}$ rising time is delayed by C_1 R_1 during power up, so that the MCU can be reset normally.



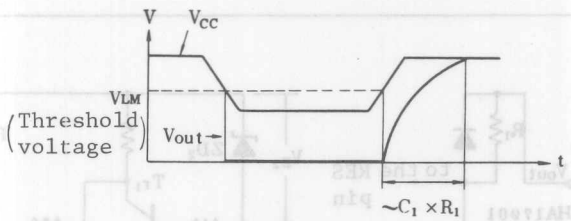
(ii) Auto-reset

The auto-reset operation of the basic circuits (a, b) in Fig. 4-4 is performed as follows.

(a) The voltages V_{Z1} and V_{F1}, applying to ZD_1 and D_1, are constant even if V_{CC} changes. The $\overline{RES}$ signals are as follows.

$$\begin{cases} V_{out} = V_{CC} > V_{Z1} + V_{F1} \\ V_{out} = GND \text{ when } V_{CC} < V_{Z1} + V_{F1} \end{cases}$$

(b) The voltage V_{Z2}, applying to ZD_2, and the ON level V_{F2} of Tr_1 are constant, even if the V_{CC} changes. The followings describe the $\overline{RES}$ signals.



$$\begin{cases} V_{out} = V_{CC} \text{ when } V_{CC} > V_{Z2} + V_{F2} \\ V_{out} = GND \text{ when } V_{CC} < V_{Z2} + V_{F2} \end{cases}$$

When the V_{CC} changes, threshold level V_{LM} is affected by replacing the Zener diodes ZD_1 and ZD_2.

(iii) The Auto-reset function will be more stable by feeding back the output signal to the $\overline{RES}$ terminal, fine tuning the reference voltage, and providing hysteresis with the V_{LM1} and the V_{LM2}. (V_{LM1} is a voltage during the shifting

from operation to resetting, and V_{LM2} is the voltage in recovering.)

4.4 Manual Reset Circuit

(1) Purpose

When the MCU is reset by an external switch, it is necessary to prevent the MCU from chattering. The following describes a reset circuit in which chattering prevention and power-on reset functions are provided.

(2) Basic Circuit

Fig. 4-5 shows the basic circuit of manual reset.

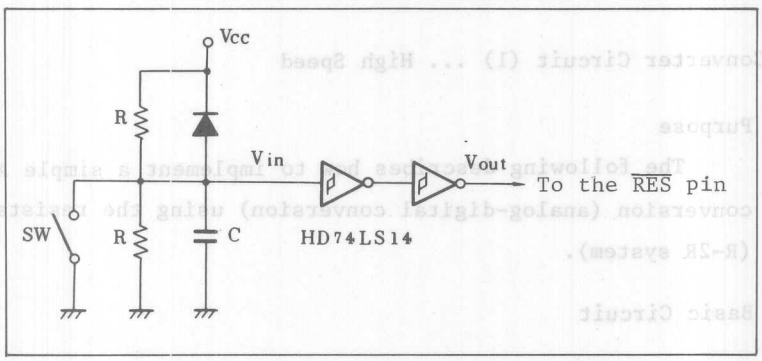
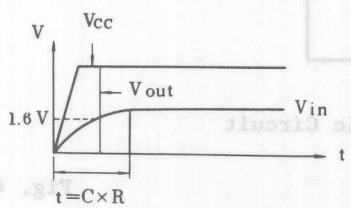


Fig. 4-5 Basic Circuit of Manual Reset

(3) Operation Principle

(i) Power-on reset

During power-up, the capacitor C will be charged and the V_{in} rising time will be delayed by the CR charge time, so that power-on would be reset normally.

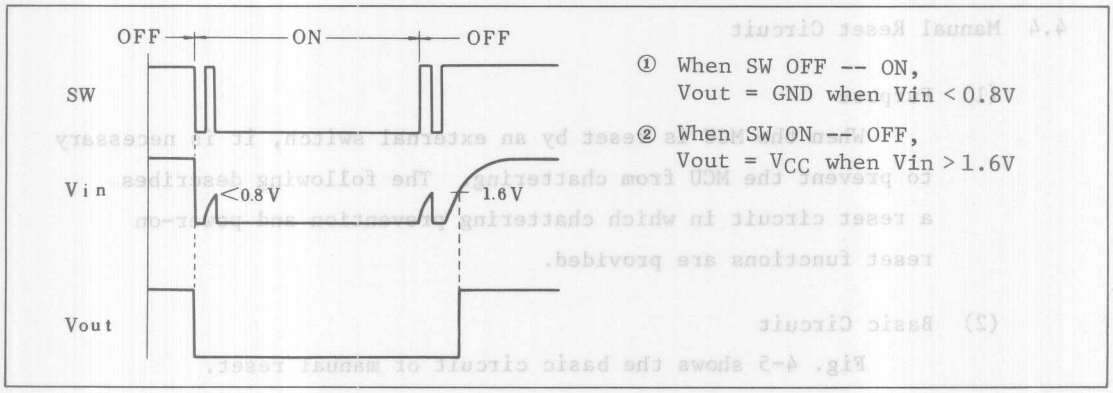


(ii) Manual reset

When SW is on, the V_{out} goes "0" and the MCU is reset

When SW is off, the capacitor C is charged, so that the rising time of V_{in} is delayed by the CR charge time.

Chattering is prevented by the capacitor C and the schmidt trigger HD74LS14.



4.5 A/D Converter Circuit (1) ... High Speed

(1) Purpose

The following describes how to implement a simple A/D conversion (analog-digital conversion) using the resistance radder (R-2R system).

(2) Basic Circuit

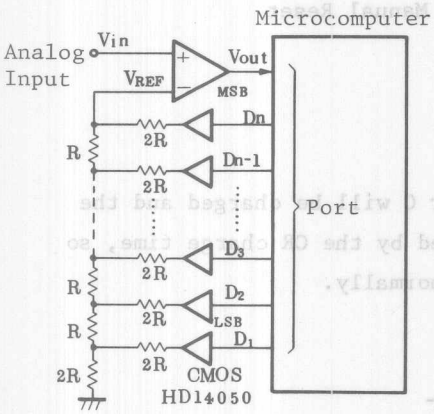


Fig. 4-6 A/D Converter Basic Circuit

$$V_{REF}=\left(\frac{Dn}{2^1}+\frac{Dn-1}{2^2}+\cdots+\frac{D_2}{2^{n-1}}+\frac{D_1}{2^n}\right)\times V$$

V : Voltage of the output "1" of the port

Vout = 0 : Vin ≦ VREF

Vout = 1 : Vin > VREF

Necessary number of port=Accuracy bit+1

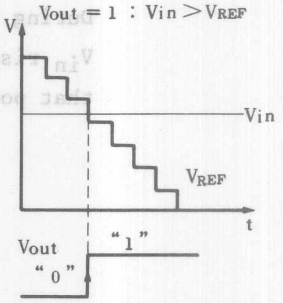


Fig. 4-7 Timing Chart

(3) Operation Outline

The following describes the operation outline of 3-bits A/D converter circuit.

(i) The digital outputs (D_3, D_2, D_1) are changed from (1, 1, 1) to (0, 0, 0) at the port. V_{REF} is reduced from 7/8 V to 0 by every 1/8 V according to Table 4-1.

(ii) The output V_{out}, which is the result of comparison between the analog input V_{in} and V_{REF}, is reversed from "0" to "1" when V_{in} is greater than V_{REF} as shown in Fig. 4-7.

(iii) The value of (D_3, D_2, D_1), when V_{out} is reversed from "0" to "1", is a digital value to the analog input V_{in}.

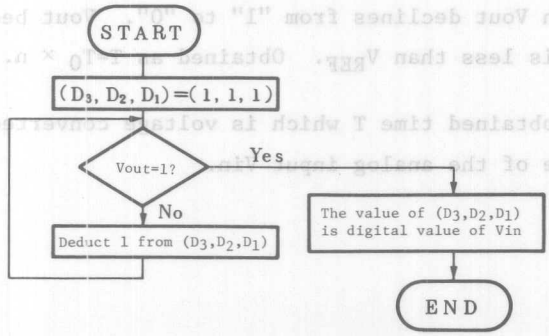
Table 4-1 Value of V_{REF}

D_3	D_2	D_1	V_{REF}
0	0	0	0
0	0	1	1/8 × V
0	1	0	2/8 × V
0	1	1	3/8 × V
1	0	0	4/8 × V
1	0	1	5/8 × V
1	1	0	6/8 × V
1	1	1	7/8 × V

$$V_{REF} = \left(\frac{D_3}{2^1} + \frac{D_2}{2^2} + \frac{D_1}{2^3}\right) \times V$$

$D_1 \sim D_3$ are denoted by "1" or "0"

V: Voltage of the port output "1"



4.6 A/D Converter Circuit (2) ... Low Speed

(1) Purpose

The following describes the A/D conversion system utilizing time for charge/discharge to/from the CR circuit. This conversion system is available even if the A/D conversion time is slow.

(2) Basic Circuit

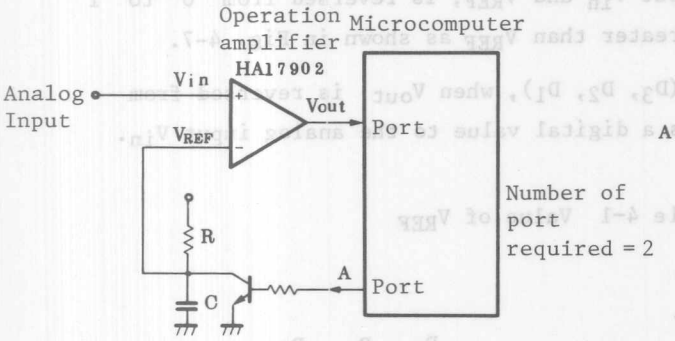


Fig. 4-8 A/D Converter Basic Circuit

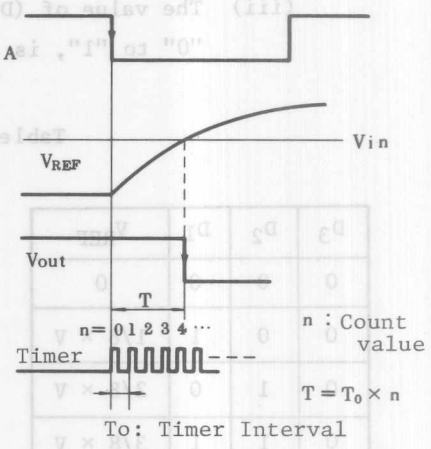


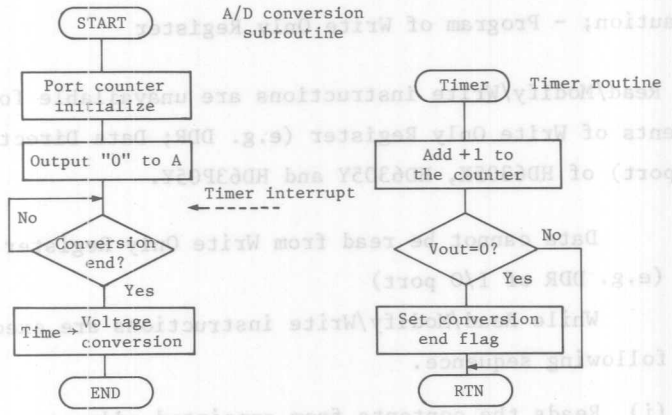
Fig. 4-9 Timing Chart

(3) Operation Outline

(i) Set the output terminal A of the port to "1", discharge the external condenser C for a sufficiently long period, decline A from "1" to "0" synchronously with the timer interrupt and charge C.

(ii) Check Vout for each timer interrupt and observe the time T in which Vout declines from "1" to "0". Vout becomes "0" when Vin is less than V_{REF}. Obtained as $T=T_0 \times n$.

(iii) The obtained time T which is voltage converted is a digital value of the analog input Vin.



4.7 Precaution;- Board Design of Oscillation Circuit

When connecting crystal and ceramic resonator with the XTAL and EXTAL pins to oscillate, observe the followings in designing the board.

(1) Locate crystal, ceramic resonator, and load capacity C1 and C2 as near the LSI as possible. (Induction of noise from outside to the XTAL and EXTAL pins may cause trouble in oscillation.)

(2) Wire the signal lines to the neighbouring XTAL and EXTAL pins as far apart as possible.

(3) Board design of situating signal lines or power supply lines near the oscillator circuit as shown in Fig. 4-11, should not be used because of trouble in oscillation by induction. The resistor between the XTAL and EXTAL, and pins close to them should be 10MΩ or more.

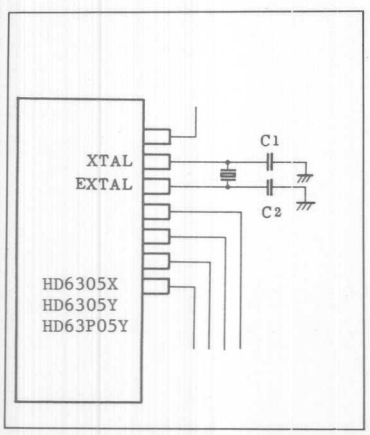


Fig. 4-10 Design of Oscillation Circuit Board

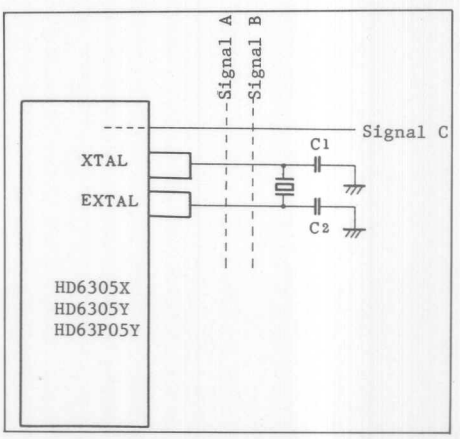


Fig. 4-11 Example Circuit Causing Trouble in Oscillation

4.8 Precaution; - Program of Write Only Register

Read/Modify/Write instructions are unavailable for changing the contents of Write Only Register (e.g. DDR; Data Direction Register of I/O port) of HD6305X, HD6305Y and HD63P05Y.

(1) Data cannot be read from Write Only Register.
(e.g. DDR of I/O port)

While Read/Modify/Write instructions are executed in the following sequence.

(i) Reads the contents from appointed address.

(ii) Changes the data which has been read.

(iii) Turn the data back to the original address.

Thus, Read/Modify/Write instructions cannot be applied to Write Only Register such as DDR.

(2) For the same reason, do not set DDR of I/O port using BSET and BCLR instructions.

(3) Stored instructions (e.g. STA and STX, etc.) are available for writting into the Write Only Register.

4.9 Precaution. - Sending/Receiving Program of Serial Data

Reading from or Writing into the SCI data register (SDR:$0012) during sending/receiving of serial data may make sending/receiving operation of SCI out of order.

4.10 Precaution; - WAIT/STOP Instructions Program

When I bit of condition code register is "1" and interrupt ($\overline{INT}$, TIMER/$\overline{INT_2}$, SCI/TIMER2) is held, the MCU does not enter into WAIT mode by executing WAIT instruction.

In that case, after the 4 dummy cycles, the MCU executes the next instruction.

In the same way, when external interrupts ($\overline{INT}$, $\overline{INT_2}$) are held at the bit I set, the MCU does not enter into the STOP mode by executing STOP instruction. In that case the MCU executes the next instruction after the 4 dummy cycles.

4.11 Precaution; - To use the EPROM ON-PACKAGE 8-bit Single-chip Microcomputer

Please be careful of the following, since this MCU has a special structure with pin socket on the package.

(1) Do not apply high static voltage or surge voltage over MAXIMUM RATINGS to the socket pins as well as the LSI pins. Otherwise permanent damage to the device would be caused.

(2) When using 32k EPROM (24-pin), insert it leaving the four pins above open.

(3) When inserting this into system products like mask ROM type single chip microcomputer, be careful of the following to give effective contact between the EPROM pins and socket pins.

(a) When soldering the LSI onto a printed circuit board, the recommended condition is:

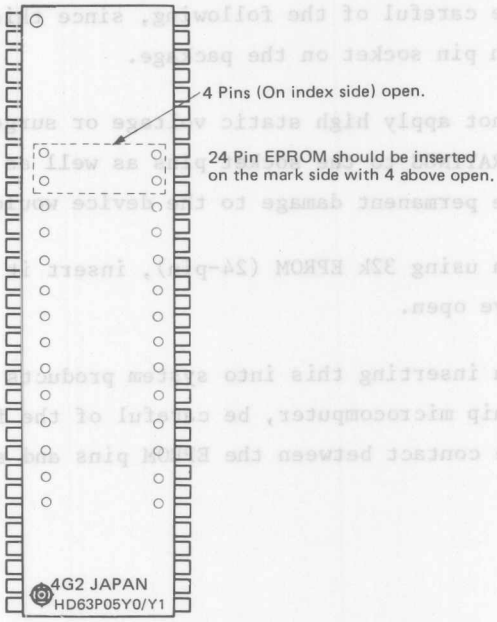
Temperature: lower than 250°C

Time: within 10 sec.

(b) Be careful that detergent or coating does not get into the socket during flux washing or board coating after soldering, because that may adversely affect the socket contact.

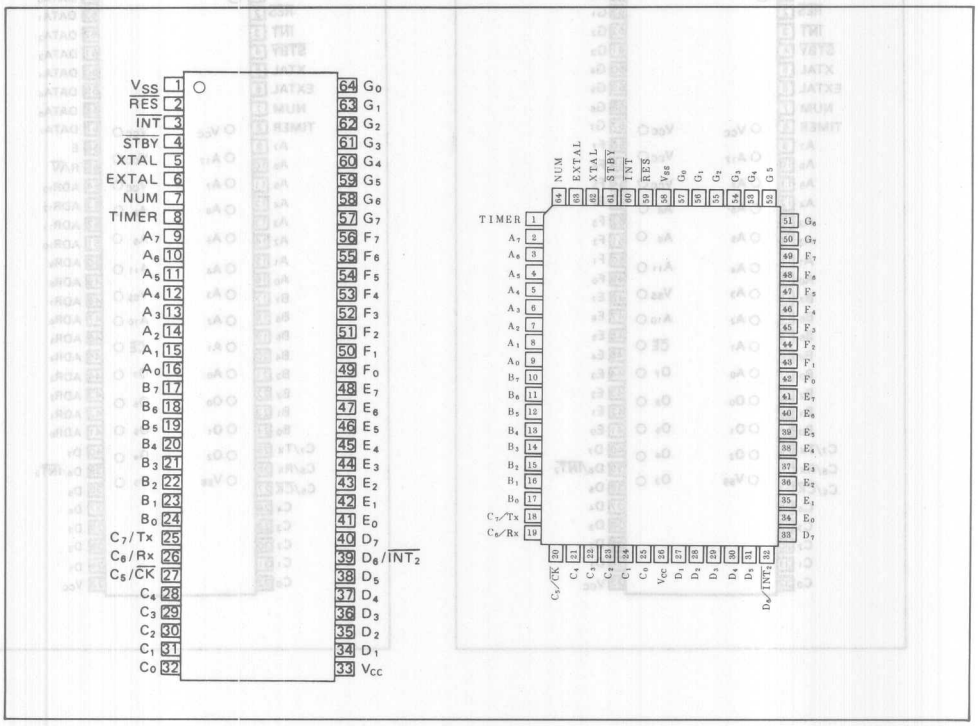
(c) Avoid permanent application of this during continuous vibration.

(d) The socket, when repeatedly inserted and removed, loses its contactability. It is recommended to use a new one when used in production.

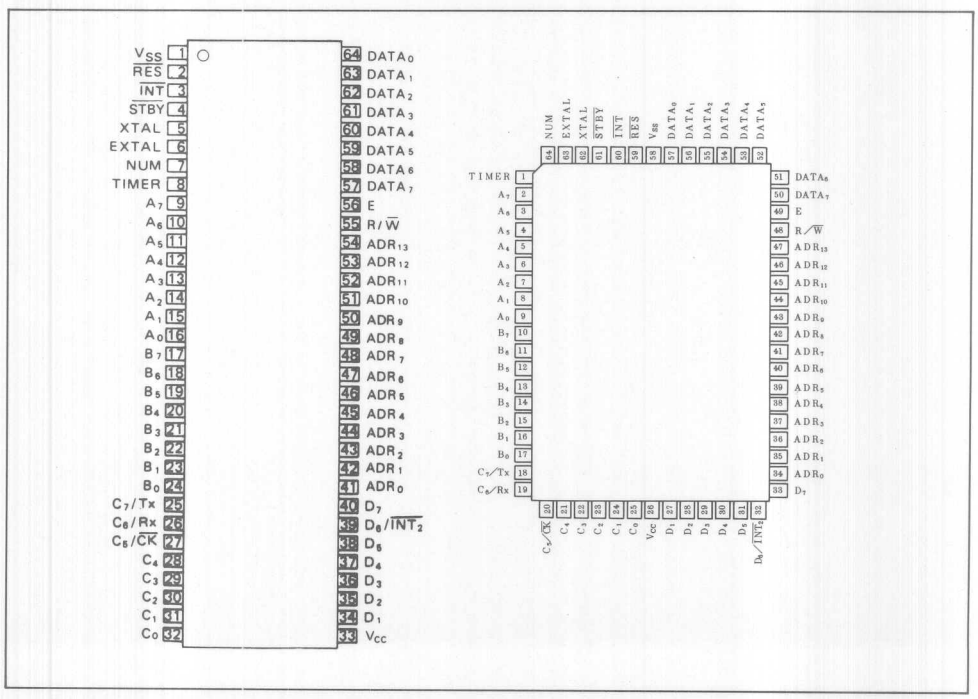


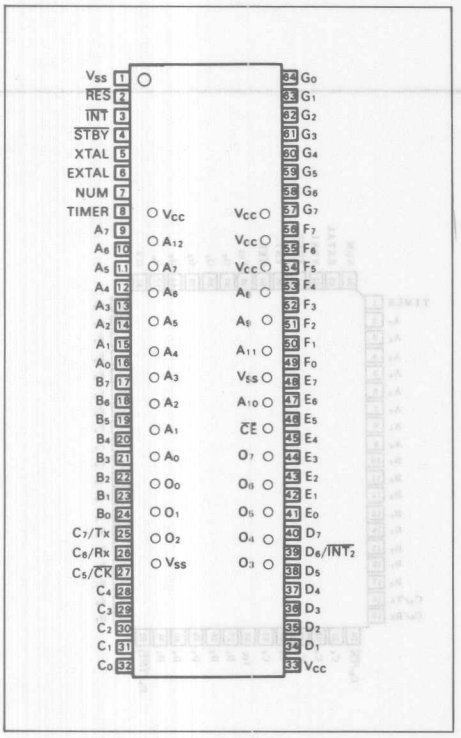
5. PIN ARRANGEMENT AND DIMENSIONAL OUTLINE

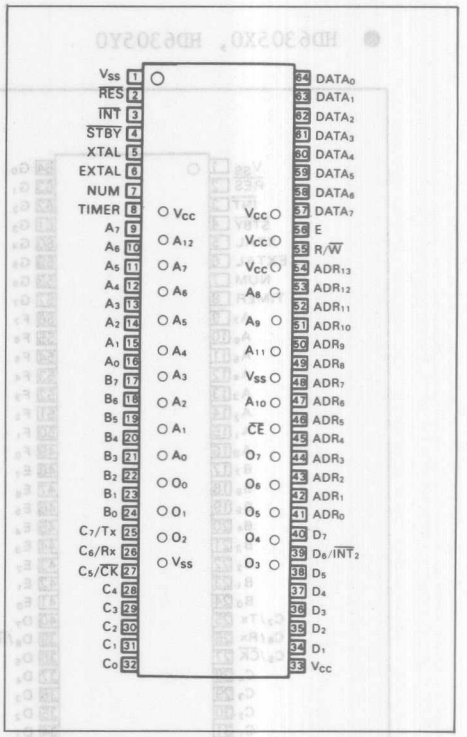
- HD6305X0, HD6305Y0



- HD6305X1, HD6305X2, HD6305Y1, HD6305Y2

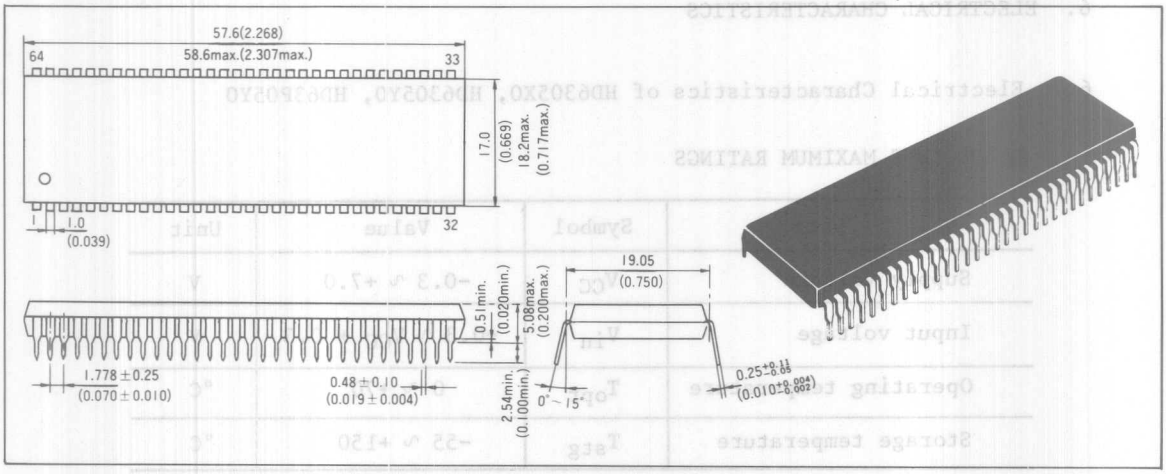



Vss 1
RES 2
INT 3
STBY 4
XTAL 5
EXTAL 6
NUM 7
TIMER 8
A7 9
A6 10
A5 11
A4 12
A3 13
A2 14
A1 15
A0 16
B7 17
B6 18
B5 19
B4 20
B3 21
B2 22
B1 23
B0 24
C7/Tx 25
C6/Rx 26
C5/CK 27
C4 28
C3 29
C2 30
C1 31
C0 32
VCC
A12
A7
A6
A5
A4
A3
A2
A1
A0
O0
O1
O2
Vss
Vcc
Vcc
Vcc
A8
A9
A11
Vss
A10
CE
O7
O6
O5
O4
O3
G0 34
G1 33
G2 32
G3 31
G4 30
G5 29
G6 28
G7 27
F7 26
F6 25
F5 24
F4 23
F3 22
F2 21
F1 20
F0 19
E7 18
E6 17
E5 16
E4 15
E3 14
E2 13
E1 12
E0 11
D7 10
D6/INT2 9
D5 8
D4 7
D3 6
D2 5
D1 4
Vcc 3

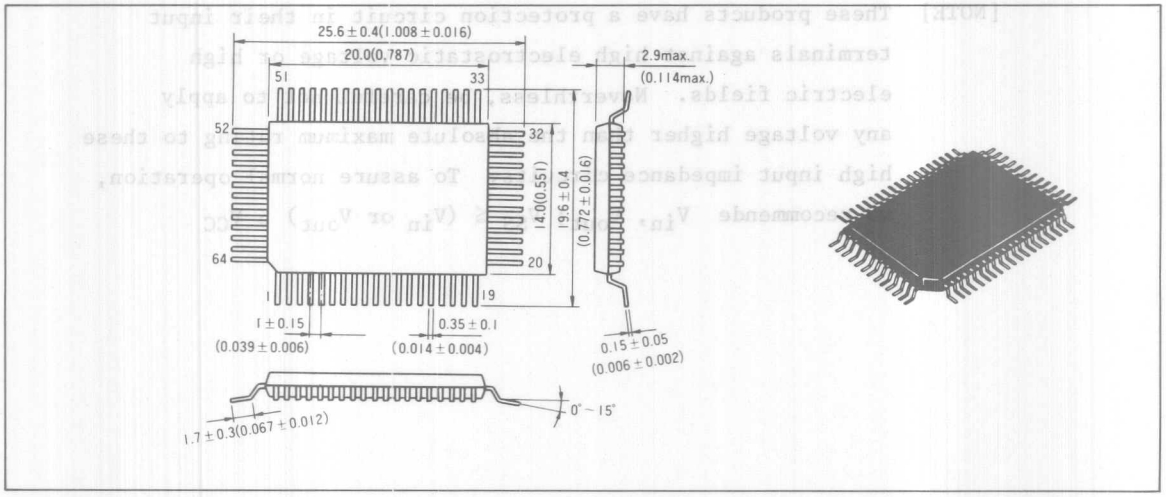

Vss 1
RES 2
INT 3
STBY 4
XTAL 5
EXTAL 6
NUM 7
TIMER 8
A7 9
A6 10
A5 11
A4 12
A3 13
A2 14
A1 15
A0 16
B7 17
B6 18
B5 19
B4 20
B3 21
B2 22
B1 23
B0 24
C7/Tx 25
C6/Rx 26
C5/CK 27
C4 28
C3 29
C2 30
C1 31
C0 32
VCC
A12
A7
A6
A5
A4
A3
A2
A1
A0
O0
O1
O2
Vss
Vcc
Vcc
Vcc
A8
A9
A11
Vss
A10
CE
O7
O6
O5
O4
O3
DATA0 34
DATA1 33
DATA2 32
DATA3 31
DATA4 30
DATA5 29
DATA6 28
DATA7 27
E 26
R/W 25
ADR13 24
ADR12 23
ADR11 22
ADR10 21
ADR9 20
ADR8 19
ADR7 18
ADR6 17
ADR5 16
ADR4 15
ADR3 14
ADR2 13
ADR1 12
ADR0 11
D7 10
D6/INT2 9
D5 8
D4 7
D3 6
D2 5
D1 4
Vcc 3

● **DP-64S**

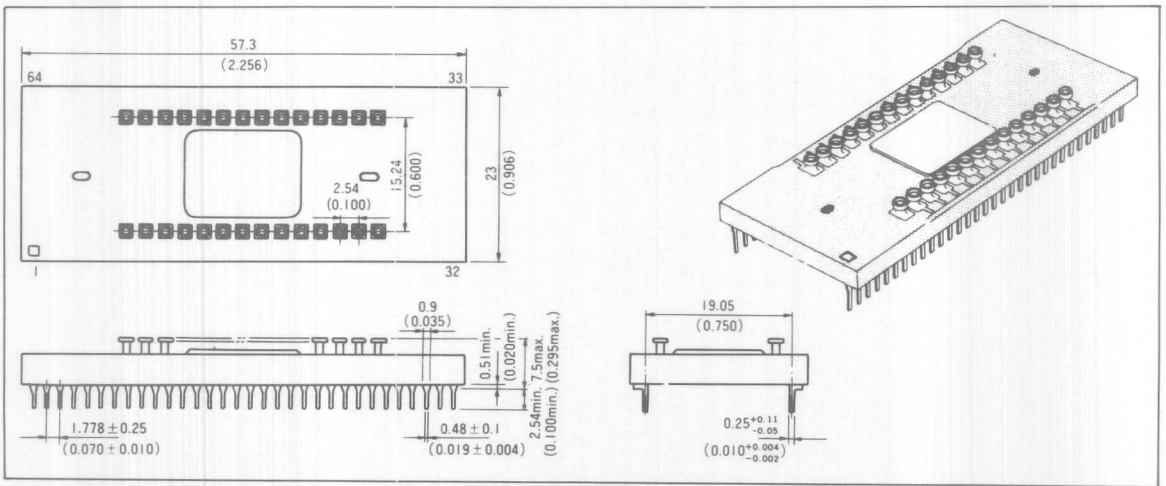
(Unit: mm(inch)



● **FP-64**



● **DC-64SP**



Note) Inch value indicated for your reference.

Fig. 5-2 Dimensional Outline

6. ELECTRICAL CHARACTERISTICS

6.1 Electrical Characteristics of HD6305X0, HD6305Y0, HD63P05Y0

■ ABSOLUTE MAXIMUM RATINGS

Item	Symbol	Value	Unit
Supply voltage	V_{CC}	-0.3 ∿ +7.0	V
Input voltage	V_{in}	-0.3 ∿ V_{CC} + 0.3	V
Operating temperature	T_{opr}	0 ∿ +70	°C
Storage temperature	T_{stg}	-55 ∿ +150	°C

[NOTE] These products have a protection circuit in their input terminals against high electrostatic voltage or high electric fields. Neverthless, be careful not to apply any voltage higher than the absolute maximum rating to these high input impedance circuits. To assure normal operation, we recommende V_{in}, V_{out}; $V_{SS} \leq (V_{in}$ or $V_{out}) \leq V_{CC}$

■ ELECTRICAL CHARACTERISTICS

● DC Characteristics (V_{CC} = 5.0V ± 10%, V_{SS} = GND and T_a = 0 ∿ +70°C unless otherwise specified)

Item		Symbol	Test condition	min	typ	max	Unit
Input voltage "High"	$\overline{RES}$, $\overline{STBY}$	V_{IH}		V_{CC}-0.5	-	V_{CC}+0.3	V
	EXTAL			V_{CC}×0.7	-	V_{CC}+0.3	V
	Others			2.0	-	V_{CC}+0.3	V
Input voltage "Low"	All Inputs	V_{IL}		-0.3	-	0.8	V
Current * dissipation	Operating	I_{CC}	f=1MHz**	-	5	10	mA
	Wait			-	2	5	mA
	Stop			-	2	10	μA
	Standby			-	2	10	μA
Input leakage current	TIMER, $\overline{INT}$, $\overline{STBY}$, D_1∿D_7	$\lvert I_{IL} \rvert$	V_{in}=0.5∿ V_{CC}-0.5V	-	-	1	μA
Three-state current	A_0 ∿ A_7, B_0 ∿ B_7, C_0 ∿ C_7, G_0 ∿ G_7, E_0 ∿ E_7,** F_0 ∿ F_7,**	$\lvert I_{TSI} \rvert$		-	-	1	μA
Input *** capacity	All terminals	C_{in}	f=1MHz, V_{in}=0V	-	-	12	pF

* The value at f=XMHz is given by

I_{CC} (f=XMHz) = I_{CC}(f=1MHz)×X. V_{IH} min = V_{CC} - 1.0V, V_{IL} max = 0.8V.

For HD63P05Y0, I_{CC} of EPROM is not included.

** In Standby Mode

*** HD63P05Y0 is MAX 15pF

● AC Characteristics (V_{CC} = 5.0V ± 10%, V_{SS} = GND, T_a = 0 ∿ +70°C, unless otherwise noted)

Item	Symbol	Test Condition	HD6305X0/Y0 HD63P05Y0			HD63A05X0/Y0 HD63PA05Y0			HD63B05X0/Y0 HD63PB05Y0			Unit
			min	typ	max	min	typ	max	min	typ	max	
Clock Frequency	f_{cl}		0.4	-	4	0.4	-	6	0.4	-	8	MHz
Cycle Time	t_{cyc}		1.0	-	10	0.666	-	10	0.5	-	10	µs
$\overline{INT}$ Pulse Width	t_{IWL}		t_{cyc} +250	-	-	t_{cyc} +200	-	-	t_{cyc} +200	-	-	ns
$\overline{INT_2}$ Pulse Width	t_{IWL2}		t_{cyc} +250	-	-	t_{cyc} +200	-	-	t_{cyc} +200	-	-	ns
$\overline{RES}$ Pulse Width	t_{RWL}		5	-	-	5	-	-	5	-	-	t_{cyc}
TIMER Pulse Width	t_{TWL}		t_{cyc} +250	-	-	t_{cyc} +200	-	-	t_{cyc} +200	-	-	ns
Oscillation Start Time (Crystal)	t_{osc}	CL=22pF±20% R_S=60Ω max	-	-	20	-	-	20	-	-	20	ms
Reset Delay Time	t_{RHL}	external capacitance 2.2µF	80	-	-	80	-	-	80	-	-	ms

● Port Characteristics (V_{CC} = 5.0V ± 10%, V_{SS} = GND, T_a = 0 ∿ +70°C, unless otherwise noted)

Item		Symbol	Test Condition	min	typ	max	Unit
Output "High" Voltage	Port A, B, C, G, E, F	V_{OH}	I_{OH} = -200µA	2.4	-	-	V
			I_{OH} = -10µA	V_{CC}-0.7	-	-	V
Output "Low" Voltage		V_{OL}	I_{OL} = 1.6mA	-	-	0.55	V
Input "High" Voltage	Port A, B, C, D, G	V_{IH}		2.0	-	V_{CC}+0.3	V
Input "Low" Voltage		V_{IL}		-0.3	-	0.8	V
Input Leakage Current		$\|I_{IL}\|$	V_{in} = 0.5 ∿ V_{CC}-0.5V	-1	-	1	µA

- SCI Timing (V_{CC} = 5.0V ± 10%, V_{SS} = GND, T_a = 0 ∿ +70°C, unless otherwise noted.)

Item	Symbol	Test Condition	HD6305X0/Y0 HD63P05Y0			HD63A05X0/Y0 HD63PA05Y0			HD63B05X0/Y0 HD63PB05Y0			Unit
			min	typ	max	min	typ	max	min	typ	max	
Clock Cylce Time	t_{Scyc}	Fig. 6-1 Fig. 6-2	1	-	32768	0.67	-	21845	0.5	-	16384	µs
Data Output Delay Time	t_{TXD}		-	-	250	-	-	250	-	-	250	ns
Data Set-up Time	t_{SRX}		200	-	-	200	-	-	200	-	-	ns
Data Hold Time	t_{HRX}		100	-	-	100	-	-	100	-	-	ns

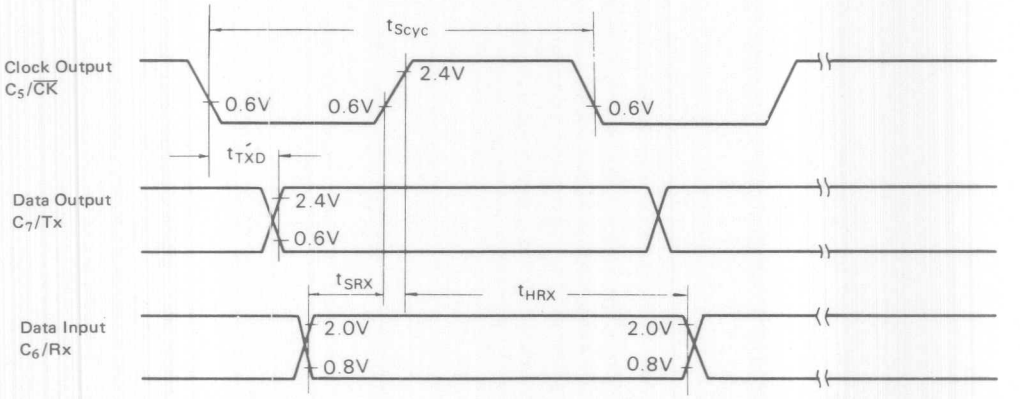


Fig. 6-1 SCI Timing (Internal Clock)

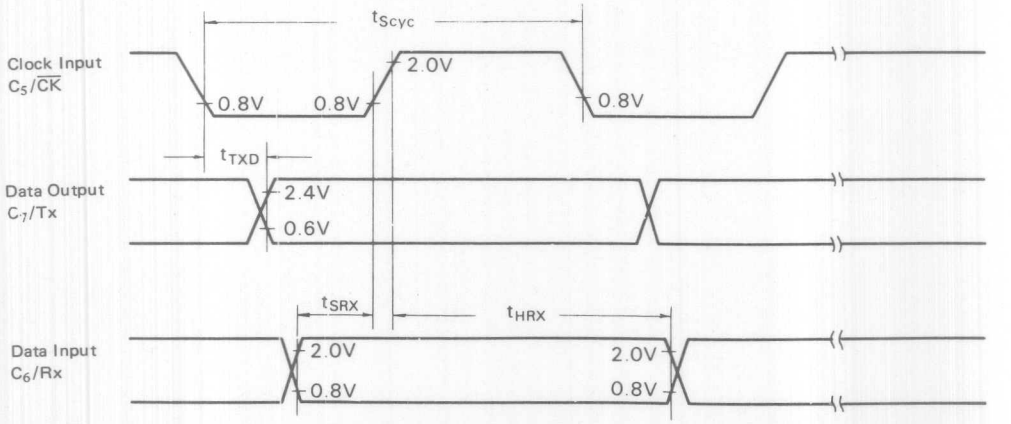
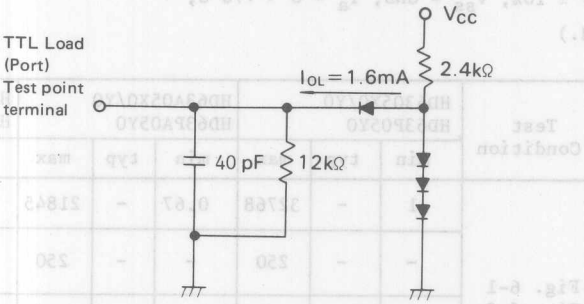


Fig. 6-2 SCI Timing (External Clock)



[NOTES] 1. The load capacitance includes stary capacitance caused by the probe, etc.
2. All diodes are 1S2074 Ⓗ.

Fig. 6-3 Test Load

6.2 Electrical Characteristics of HD6305X1/X2, HD6305Y1/Y2, HD63P05Y1

■ ABSOLUTE MAXIMUM RATINGS

Item	Symbol	Value	Unit
Supply voltage	V_{CC}	$-0.3 \sim +7.0$	V
Input voltage	V_{in}	$-0.3 \sim V_{CC} + 0.3$	V
Operating temperature	T_{opr}	$0 \sim +70$	°C
Storage temperature	T_{stg}	$-55 \sim +150$	°C

[NOTE] These products have a protection circuit in their input terminals against high electrostatic voltage or high electric fields. Notwithstanding, be careful not to apply any voltage higher than the absolute maximum rating to these high input impedance circuits. To assure normal operation, we recommended V_{in}, V_{out}; $V_{SS} \leqq (V_{in} \text{ or } V_{out}) \leqq V_{CC}$

■ ELECTRICAL CHARACTERISTICS

- DC Characteristics (V_{CC} = 5.0V ± 10%, V_{SS} = GND, T_a = 0 ∿ +70°C, unless otherwise specified.)

Item		Symbol	Test Condition	min	typ	max	Unit
Input voltage "High"	$\overline{RES}$, $\overline{STBY}$	V_{IH}		V_{CC}-0.5	-	V_{CC}+0.3	V
	EXTAL			V_{CC}×0.7	-	V_{CC}+0.3	V
	Others			2.0	-	V_{CC}+0.3	V
Input voltage "Low"	All Input	V_{IL}		-0.3	-	0.8	V
Output voltage "High"	All Output	V_{OH}	I_{OH}=-200μA	2.4	-	-	V
			I_{OH}=-10μA	V_{CC}-0.7	-	-	
Output voltage "Low"	All Output	V_{OL}	I_{OL}=1.6mA	-	-	0.55	V
Input leakage current	$\overline{TIMER}$, $\overline{INT}$, D_1 ∿ D_7, $\overline{STBY}$	$\lvert I_{IL} \rvert$	V_{in}=0.5 ∿ V_{CC} - 0.5V	-	-	1.0	μA
Three-state current	A_0 ∿ A_7, B_0 ∿ B_7, C_0 ∿ C_7, ADR_0 ∿ ADR_{13},* $DATA_0$ ∿$DATA_7$, E*, R/$\overline{W}$*	$\lvert I_{TSI} \rvert$		-	-	1.0	μA
Current** dissipation	Operating	I_{CC}	f = 1MHz***	-	5	10	mA
	Wait			-	2	5	mA
	Stop			-	2	10	μA
	Standby			-	2	10	μA
Input**** capacity	All terminals	C_{in}	f = 1MHz, V_{in} = 0V	-	-	12	pF

* At standby mode

** V_{IH} min = V_{CC} - 1.0V. V_{IL} max = 0.8 V. For HD63P05Y1, I_{CC} of EPROM is not included.

*** The value at f = xMHz can be calculated by the following equation: I_{CC} (f = xMHz) = I_{CC} (f = 1MHz) multiplied by X.

**** HD63P05Y1 is MAX. 15pF

● AC Characteristics (V_{CC} = 5.0V ± 10%, V_{ss} =GND, T_a = 0 ∿ +70°C, unless otherwise specified.)

Item	Symbol	Test Condition	HD6305X1/X2/Y1/Y2 HD63P05Y1			HD63A05X1/X2/Y1/Y2 HD63PA05Y1			HD63B05X1/X2/Y1/Y2 HD63PB05Y1			Unit
			min	typ	max	min	typ	max	min	typ	max	
Cycle Time	t_{cyc}	Fig. 6-4	1	–	10	0.666	–	10	0.5	–	10	µs
Enable Rise Time	t_{Er}		–	–	20	–	–	20	–	–	20	ns
Enable Fall Time	t_{Ef}		–	–	20	–	–	20	–	–	20	ns
Enable Pulse Width ("High" Level)	PW_{EH}		450	–	–	300	–	–	220	–	–	ns
Enable Pulse Width ("Low" Level)	PW_{EL}		450	–	–	300	–	–	220	–	–	ns
Address Delay Time	t_{AD}		–	–	250	–	–	190	–	–	180	ns
Address Hold Time	t_{AH}		40	–	–	30	–	–	20	–	–	ns
Data Delay Time	t_{DW}		–	–	200	–	–	160	–	–	120	ns
Data Hold Time (Write)	t_{HW}		40	–	–	30	–	–	20	–	–	ns
Data Set-up Time (Read)	t_{DSR}		80	–	–	60	–	–	50	–	–	ns
Data Hold Time (Read)	t_{HR}		0	–	–	0	–	–	0	–	–	ns

● Port Timing (V_{CC} = 5.0V ± 10%, V_{SS} = GND, T_a = 0 ∿ +70°C, unless otherwise noted.)

Item	Symbol	Test Condition	HD6305X1/X2/Y1/Y2 HD63P05Y1			HD63A05X1/X2/Y1/Y2 HD63PA05Y1			HD63B05X1/X2/Y1/Y2 HD63PB05Y1			Unit
			min	typ	max	min	typ	max	min	typ	max	
Port Data Set-up Time (Port A, B, C, D)	t_{PDS}	Fig. 6-5	200	–	–	200	–	–	200	–	–	ns
Port Data Hold Time (Port A, B, C, D)	t_{PDH}		200	–	–	200	–	–	200	–	–	ns
Port Data Delay Time (Port A, B, C)	t_{PDW}	Fig. 6-6	–	–	300	–	–	300	–	–	300	ns

● Control Signal Timing (V_{CC} = 5.0V ± 10%, V_{SS} = GND, T_a = 0 ∿ +70°C, unless otherwise noted.)

Item	Symbol	Test Condition	HD6305X1/X2/Y1/Y2 HD63P05Y1			HD63A05X1/X2/Y1/Y2 HD63PA05Y1			HD63B05X1/X2/Y1/Y2 HD63PB05Y1			Unit
			min	typ	max	min	typ	max	min	typ	max	
$\overline{INT}$ Pulse Width	t_{IWL}		t_{cyc} +250	–	–	t_{cyc} +200	–	–	t_{cyc} +200	–	–	ns
$\overline{INT_2}$ Pulse Width	t_{IWL2}		t_{cyc} +250	–	–	t_{cyc} +200	–	–	t_{cyc} +200	–	–	ns
$\overline{RES}$ Pulse Width	t_{RWL}		5	–	–	5	–	–	5	–	–	t_{cyc}
Control Set-up Time	t_{CS}	Fig. 6-8	250	–	–	250	–	–	250	–	–	ns
Timer Pulse Width	t_{TWL}		t_{cyc} +250	–	–	t_{cyc} +200	–	–	t_{cyc} +200	–	–	ns
Oscillation Start Time (Crystal)	t_{OSC}	Fig. 6-8*	–	–	20	–	–	20	–	–	20	ms
Reset Delay Time	t_{RHL}	**	80	–	–	80	–	–	80	–	–	ms

* C_L = 22pF±20%, R_S = 60Ω max.

** 2.2 μF

● SCI Timing (V_{CC} = 5.0V ± 10%, V_{SS} = GND, T_a = 0 ∿ +70°C, unless otherwise noted)

Item	Symbol	Test Condition	HD6305X1/X2/Y1/Y2 HD63P05Y1			HD63A05X1/X2/Y1/Y2 HD63PA05Y1			HD63B05X1/X2/Y1/Y2 HD63PB05Y1			Unit
			min	typ	max	min	typ	max	min	typ	min	
Clock Cycle Time	t_{Scyc}	Fig. 6-9 Fig. 6-10	1	-	32768	0.67	-	21845	0.5	-	16384	µs
Data Output Delay Time	t_{TXD}		-	-	250	-	-	250	-	-	250	ns
Data Set-up Time	t_{SRX}		200	-	-	200	-	-	200	-	-	ns
Data Hold Time	t_{HRX}		100	-	-	100	-	-	100	-	-	ns

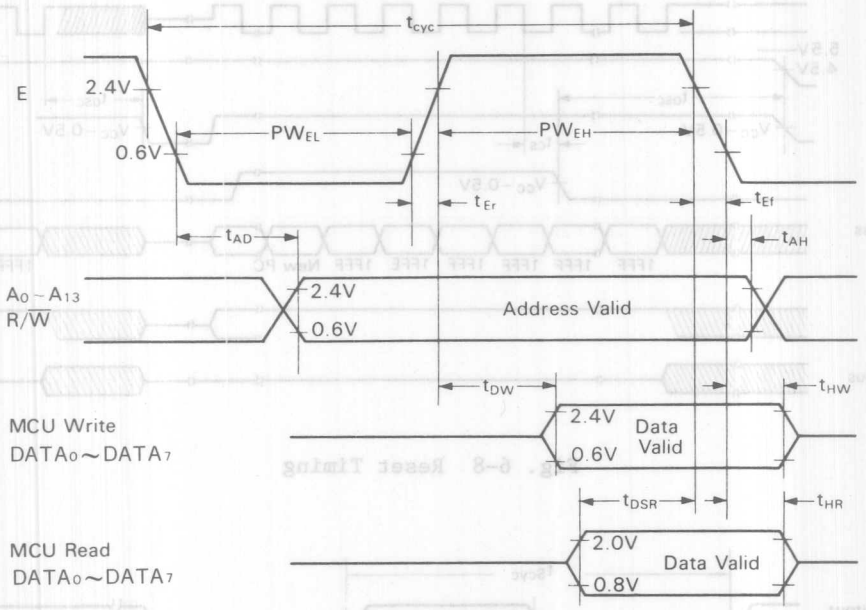


Fig. 6-4 Bus Timing

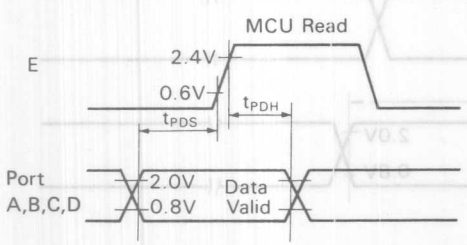


Fig. 6-5 Port Data Set-up and Hold Times (MCU Read)

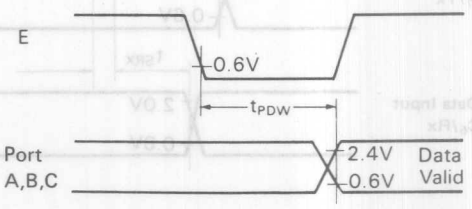


Fig. 6-6 Port Data Delay Time (MCU Write)

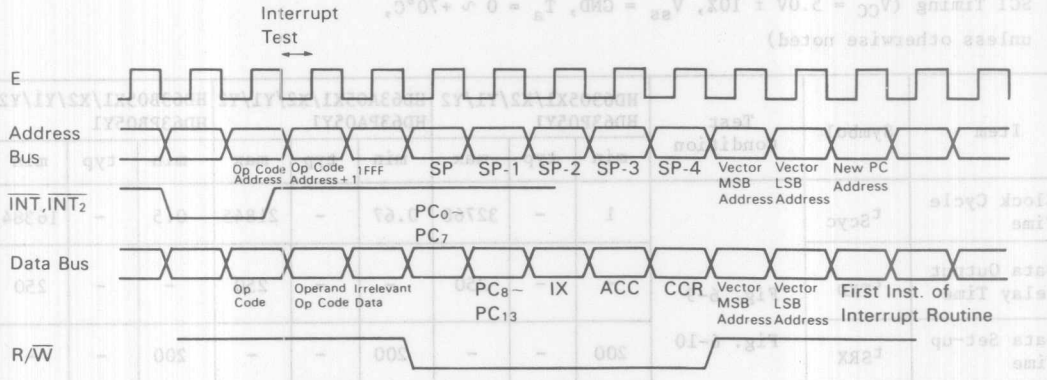


Fig. 6-7 Interrupt Sequence

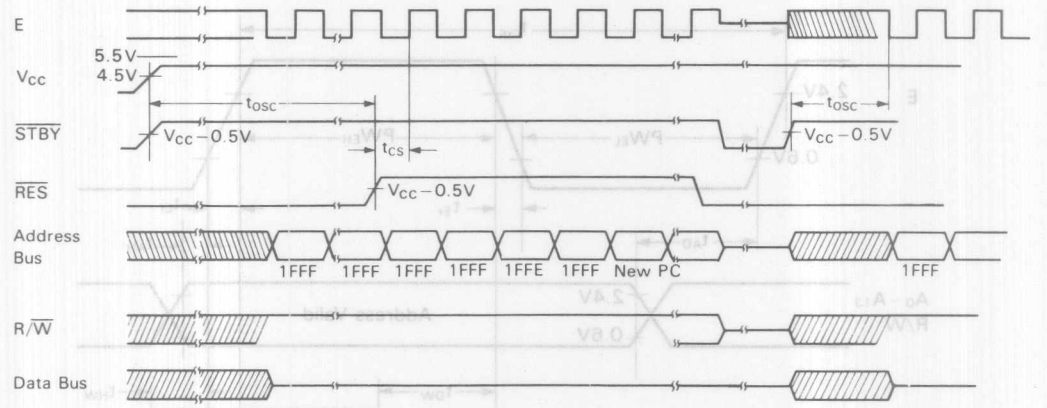


Fig. 6-8 Reset Timing

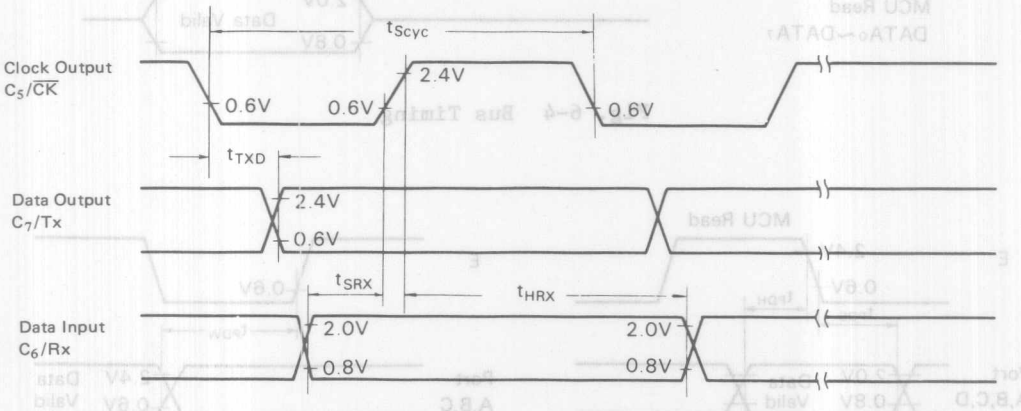


Fig. 6-9 SCI Timing (Internal Clock)

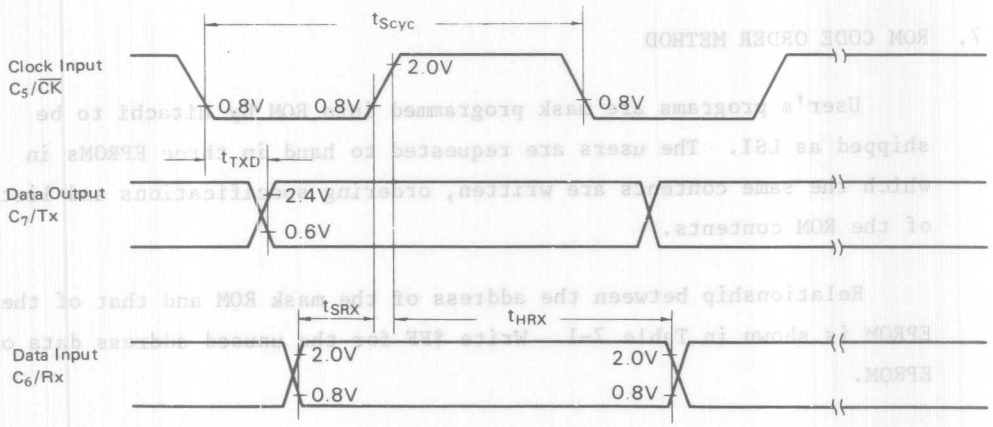
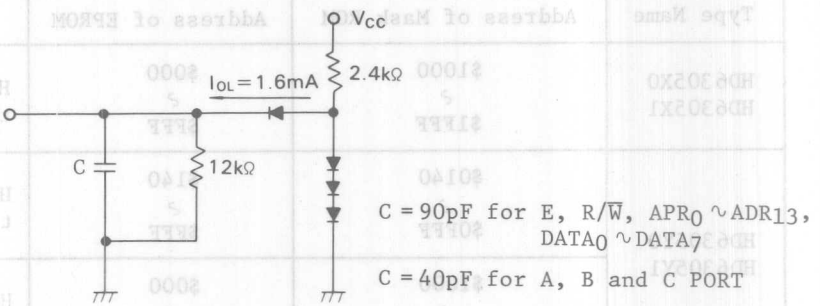


Fig. 6-10 SCI Timing (External Clock)



[NOTES] 1. The load capacitance includes stary capacitance caused by the probe, etc.
2. All diodes are 1S2074 Ⓗ.

Fig. 6-11 Test Load

7. ROM CODE ORDER METHOD

User's programs are mask programmed into ROM by Hitachi to be shipped as LSI. The users are requested to hand in three EPROMs in which the same contents are written, ordering specifications and list of the ROM contents.

Relationship between the address of the mask ROM and that of the EPROM is shown in Table 7-1. Write $FF for the unused address data of EPROM.

Table 7-1 Relationship between the Address of Mask ROM and that of EPROM

Type Name	Address of Mask ROM	Address of EPROM	Remarks
HD6305X0 HD6305X1	$1000 ≀ $1FFF	$000 ≀ $FFF	HN482732A or their equivalent
HD6305Y0 HD6305Y1	$0140 ≀ $0FFF	$140 ≀ $FFF	HN482732A or their equivalent
	$1000 ≀ $1FFF	$000 ≀ $FFF	HN482732A or their equivalent
HD6305Y0 HD6305Y1	$0140 ≀ $1FFF	$0140 ≀ $1FFF	HN482764 or their equivalent

APPENDIX

I. DESIGN PROCEDURE AND SUPPORT TOOL

Cross assembler and Hardware Simulator, containing various kinds of computers, are available as supporting systems to develop user's programs. Hitachi will mask program user's programs into ROM to ship them as LSI.

Fig. I-1 shows a typical program design procedure and Table I-1 summarizes a set of system development supporting tool for the HD6305 MCU.

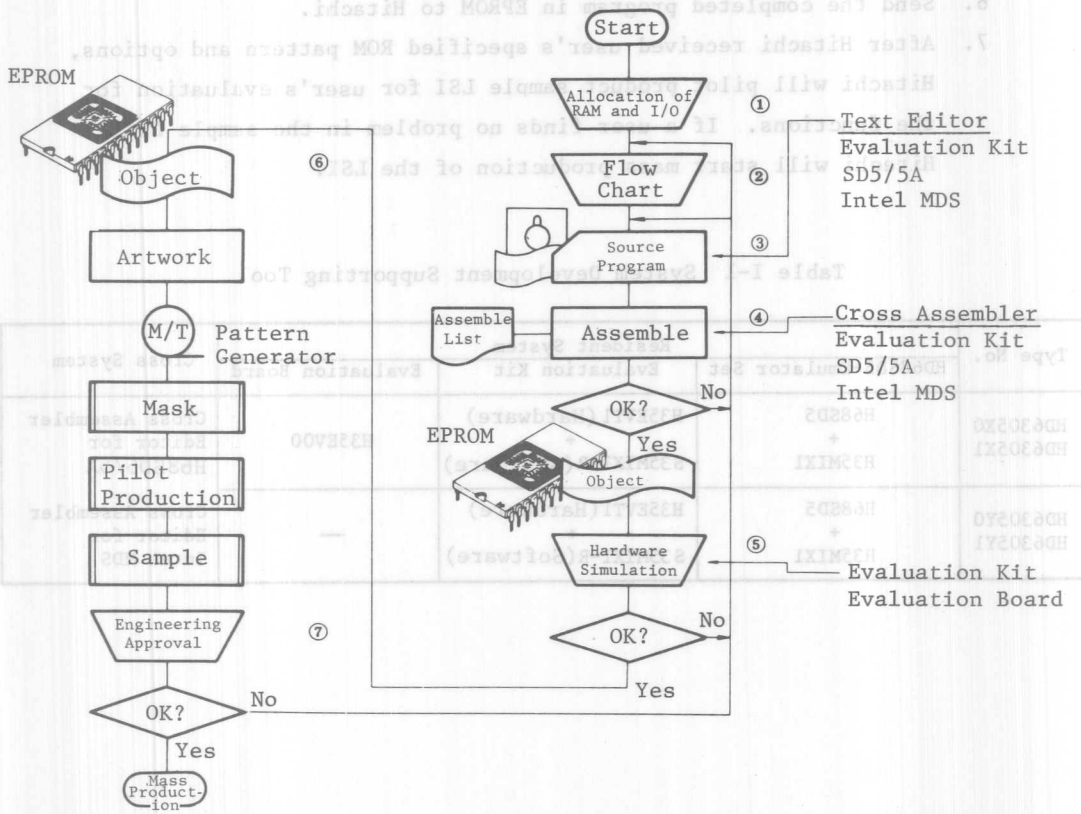


Fig. I-1 Program Design Procedure

1. Specify functional assignment of I/O pins and allocation of RAM area before starting programming.
2. Design flowchart to implement the functions and encode this flowchart with mnenic codes.
3. Punch the coded format onto cards or paper tape, or write it on a floppy disk. This set of coded form is a source program.
4. Assemble the source program to form an object program with an assembler on either a resident system (evaluation kit) or cross system. Then check errors out.
5. Verify the program through hardware simulation by an aid of evaluation kit or evaluation board.
6. Send the completed program in EPROM to Hitachi.
7. After Hitachi received user's specified ROM pattern and options, Hitachi will pilot product sample LSI for user's evaluation for the functions. If a user finds no problem in the sample LSI, Hitachi will start mass production of the LSI.

Table I-1 System Development Supporting Tool

Type No.	Resident System			Cross System
	HD68SD5+Emulator Set	Evaluation Kit	Evaluation Board	
HD6305X0 HD6305X1	H68SD5 + H35MIX1	H35EVT1(Hardware) + S35MIX1-R(Software)	H35EV00	Cross Assembler Editor for H68SD5/5A
HD6305Y0 HD6305Y1	H68SD5 + H35MIX1	H35EVT1(Hardware) + S35MIX1-R(Software)	—	Cross Assembler Editor for Intel MDS

II. ORDERING SPECIFICATIONS

(1) Basic ITEM (Please fill in blank space or mark applicable item with symbol)

LSI Family		Outline	
Application		Options	
Customer ROM Code ID.			
ROM Code Media	EPROM		

(2) Environment Check List

LSI Ambient Temperature	nominal	°C	Target Level of Reliability	500 fit 1000 fit
	range	°C- °C		
LSI Ambient Humidity	nominal	%	Acceptable Quality Level	1.0% 0.65% 0.4%
	range	%- %		
Power ON Duration		hour/day typ.	Remarks	
Maximum Applied Voltage to LSI		V		

(3) Electrical Characteristics

Purchasing Specifications	Hitachi's Standard Specifications
_____	Refer to Data Sheet _____

Please fill in the space enclosed with ☐

ROM Code Verification

LSI Type No.	
Shipping Date of ROM to Customers	
Approved Date of ROM from Customers	

Date of Order _____

Customer _____

Dept. _____

Accepted by _____

II. ORDERING SPECIFICATIONS

(1) Basic ITEM (Please fill in blank space or mark applicable item with symbol ○)

LSI Family		Outline	
Application		Options	
Customer ROM Code ID.			
ROM Code Media	EPROM		

(2) Environment Check List

LSI Ambient Temperature	nominal	°C		Target Level of Reliability	500 fit 1000 fit
	range	°C ~ °C			
LSI Ambient Humidity	nominal	%		Acceptable Quality Level	1.0% 0.65% 0.4%
	range	% ~ %			
Power ON Duration		hour/day typ.		Remarks	
Maximum Applied Voltage to LSI		V			

(3) Electrical Characteristics

Purchasing Specifications _____	Hitachi's Standard Specifications Refer to Data Sheet _____

Please fill in the space enclosed with ☐

ROM Code Verification

LSI Type No.	
Shipping Date of ROM to Customers	
Approved Date of ROM from Customers	

Date of Order _____

Customer _____

Dept. _____

Accepted by _____